

The Online Linguistic Database

Software for Linguistic Fieldwork

by

Joel Robert William Dunham

B.A., The University of British Columbia, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Linguistics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

December 2014

© Joel Robert William Dunham 2014

Abstract

The documentation and analysis of endangered languages is a core component of the linguistic endeavour. Language consultants and linguistic researchers collaborate to generate a variety of data which in turn fuel theoretical discovery and language revitalization. This dissertation describes and evaluates a piece of software designed to facilitate new, and enhance existing, collaboration, documentation, and analysis. But beyond this, it argues for the value of a certain methodological approach to linguistics broadly construed, one in which computation is key and where provisions are made for collaboration, data-sharing and data reuse.

The Online Linguistic Database (OLD) is open source software for creating web applications that facilitate collaborative linguistic fieldwork. The OLD allows fieldworkers to continue doing what they are already doing—eliciting, transcribing, recording, and analyzing forms and creating data sets and papers with them—but collaboratively. This point should not be understated: though practises are changing, linguistic fieldwork currently involves a loose network of relatively isolated practitioners and data sets; simply creating the infrastructure for collaboration and data-sharing is half the battle. The other half is creating features and conveniences that make the software worth using. In this domain, the OLD provides automated feedback on lexi-

Abstract

cal consistency of morphological analyses, sophisticated search, the creation and (structural) searching of arbitrarily many corpora and treebanks, and the specification and computational implementation of models of the lexicon, phonology, and morphology, upon which are built practical morphological parsers.

The dissertation describes the OLD, motivating its design decisions and arguing that it has the potential to contribute positively to the achievement of the three core goals of linguistic fieldwork, namely documentation, research, and language revitalization. Particular attention is paid to the practical and research-related advantages of the morphophonological modelling capability with examples and evaluations of morphological parsers created for the Blackfoot language.

Preface

This dissertation consists of original and independent work by the author, Joel Dunham. The data set described in chapter four contains data gathered during fieldwork with speakers of the Blackfoot language which was carried out by the author under UBC Ethics Certificate number H10-02768 as well as by a number of other researchers whose contributions are acknowledged at relevant locations in the text. The content of this dissertation is summarized in Dunham, J., Cook, G., and Horner, J. (2014), *LingSync & the Online Linguistic Database: New models for the collection and management of data for language communities, linguists and language learners*, *ACL 2014*. Apart from the small amount of overlap with that publication, this dissertation consists of wholly unpublished work.

Table of Contents

Abstract	ii
Preface	iv
Table of Contents	v
List of Tables	ix
List of Figures	x
List of Acronyms	xiii
Acknowledgements	xv
Dedication	xviii
1 Introduction	1
1.1 The argument	4
1.1.1 Endangered languages are valuable	5
1.1.2 Fieldwork	9
1.1.3 Better fieldwork	13
1.1.4 The Online Linguistic Database	15
1.2 Summary	17
2 Fieldwork	19
2.1 Overview of the OLD	21
2.1.1 History	21
2.1.2 Strategy	25
2.1.3 Description	26
2.1.4 Implementation	35
2.2 Other fieldwork tools	37
2.2.1 SIL International	40
2.2.2 Toolbox	44

Table of Contents

2.2.3	FLEx	50
2.2.4	LingSync	70
2.2.5	Summary	99
2.3	Features	101
2.3.1	Sharing data	103
2.3.2	Architecture	108
2.3.3	Open source & web-based	112
2.3.4	Data structure	113
2.3.5	Interface	123
2.3.6	Search	127
2.3.7	Automation	140
2.3.8	Consistency	141
2.3.9	Dictionary	147
2.3.10	Documents	148
2.3.11	Access	152
2.3.12	Documentation	156
2.4	Summary	158
3	Morphological Parser Creator	159
3.1	Architecture	160
3.2	Finite-state machines & grammars	164
3.3	Phonologies	170
3.4	Morphologies	176
3.5	Morphophonologies	186
3.6	<i>N</i> -gram language models	192
3.7	Summary	202
4	Blackfoot	205
4.1	Description	205
4.2	The Blackfoot data set	213
4.2.1	Morphemes	217
4.2.2	Morphologically well analyzed words	225
4.3	Issues in the data set	232
4.3.1	Morpheme categorization	233
4.3.2	Variability in morphological analyses	235
4.3.3	Orthographic inconsistencies	238
4.4	Summary	242

Table of Contents

5	Morphological Parsers for Blackfoot	244
5.1	Parser creation and evaluation procedures	246
5.2	Parser 1	249
5.2.1	Phonology	250
5.2.2	Morphology	286
5.2.3	Language Model	290
5.2.4	Results	292
5.2.5	Discussion	293
5.3	Parser 2	297
5.3.1	Phonology	298
5.3.2	Morphology	299
5.3.3	Language Model	299
5.3.4	Results	302
5.3.5	Discussion	303
5.4	Discussion	309
6	Conclusion	312
	Bibliography	320

Appendix

A	OLD 1.0 Data Structure	337
A.1	Relational databases	338
A.2	OLD objects	340
A.2.1	Application settings	341
A.2.2	Collection	349
A.2.3	Collection backup	358
A.2.4	Elicitation method	359
A.2.5	File	360
A.2.6	Form	367
A.2.7	Form backup	376
A.2.8	Form search	377
A.2.9	Language	378
A.2.10	Orthography	379
A.2.11	Page	380
A.2.12	Source	382
A.2.13	Speaker	392
A.2.14	Syntactic category	393

Table of Contents

A.2.15 Tag	394
A.2.16 Translation	396
A.2.17 User	396

List of Tables

2.1	Data in OLD applications (Feb 14, 2014)	24
2.2	Comparison of fieldwork software.	39
2.3	Features across OLD versions	102
3.1	Morphology types.	178
4.1	Morpheme categories of the Blackfoot dictionary (Frantz and Russell, 1995).	213
4.2	Sources of the Blackfoot data set.	214
4.3	Forms in the Blackfoot data set, grouped by category.	217
4.4	Morphologically well analyzed word sets in the Blackfoot data set.	227
4.5	20 most common well analyzed words in the Blackfoot data set.	229
4.6	Ten most common well analyzed <i>verbals</i> in the Blackfoot data set.	230
4.7	Ten most common well analyzed <i>nominals</i> in the Blackfoot data set.	231
4.8	20 most common category strings of well analyzed words in the Blackfoot data set.	232
5.1	20 most frequent trigrams in one of Parser 1’s training language models (LMs).	291
5.2	Parser 1 results.	292
5.3	20 most frequent category trigrams in one of Parser 2’s training LMs.	301
5.4	Parser 2 results.	302
A.1	BibTex source types and required attributes.	391

List of Figures

2.1	Dictionary-style representations targeted by FLE _x (cf. SIL International, 2014a).	54
2.2	OLD UML diagram.	109
2.3	Screen shot of the Blackfoot OLD interface showing an IGT form display.	124
3.1	Schema of an OLD morphological parser.	161
3.2	Finite-state automaton (FSA) network diagram for $a b^* a$	166
3.3	Network diagram for $\mathbf{a} \rightarrow \mathbf{b} \mid \backslash \mathbf{b} _$	169
3.4	Finite-state transducer (FST) network diagram for $\text{"-"} \rightarrow \mathbf{s} \mid \mathbf{k} _ \mathbf{I} ;$	173
3.5	Partial Blackfoot phonology.	174
3.6	Partial Blackfoot phonology with test declarations.	176
3.7	Morphology rewrite rule script examples.	181
3.8	Context-free (CF) grammar for a simple morphology.	182
3.9	Network diagram for $\mathbf{l} \mathbf{a} \mid \mathbf{l} \mathbf{e}$	182
3.10	Network diagram for $\mathbf{l} \mathbf{a} \mathbf{0}:\text{" the D"} \mid \mathbf{l} \mathbf{e} \mathbf{0}:\text{" the D"}$	183
3.11	Network diagram for $\mathbf{l} \mathbf{a} \% \mathbf{t} \mathbf{h} \mathbf{e} \% \mathbf{D} \mid \mathbf{l} \mathbf{e} \% \mathbf{t} \mathbf{h} \mathbf{e} \% \mathbf{D}$	184
3.12	Lexc morphology scripts.	185
3.13	Toy French phonology.	187
3.14	Toy French morphophonology.	188
3.15	Network diagram for the toy French morphophonology.	190
3.16	Toy French morphophonology with <i>rurl</i> morphology.	191
3.17	Simple LM corpus file (first four lines, repeated 100 times).	198
3.18	MITLM-generated ARPA LM file: $\text{order} = 3$, $\text{smoothing} = \text{ModKN}$	199
4.1	Blackfoot phoneme inventory (IPA).	206
4.2	Blackfoot phoneme inventory (orthographic).	207
4.3	Blackfoot vowel phones.	209
5.1	Parser 1 phonology: ordered phonological rules.	254

List of Figures

5.2	Phoneme class regular expressions.	255
5.3	Coalescence.	256
5.4	Semivowel loss.	256
5.5	Gemination.	257
5.6	s-connection.	259
5.7	y-reduction.	259
5.8	Breaking.	260
5.9	o-replacement.	260
5.10	ih-loss.	261
5.11	Presibilation.	261
5.12	sss-shortening.	261
5.13	Semivowel drop.	262
5.14	Vowel shortening.	262
5.15	t-affrication.	262
5.16	Postsibilation.	263
5.17	i-absorption.	263
5.18	Desyllabification.	264
5.19	Glottal metathesis.	265
5.20	Vowel epenthesis.	266
5.21	Glottal reduction.	266
5.22	Glottal loss.	267
5.23	Glottal assimilation.	267
5.24	Accent spread.	268
5.25	Break delete.	268
5.26	i-loss.	269
5.27	Parser 1 phonology: lexical phonological rules.	270
5.28	Irregular breaking.	271
5.29	Semivowel alternation.	272
5.30	Nasal-initial verbs.	274
5.31	Initial vowel elision in imperatives.	275
5.32	Codafication.	275
5.33	Stop-stop epenthesis.	276
5.34	Become /o/.	277
5.35	Become /a/.	277
5.36	Nasal loss.	277
5.37	Variable-length vowels.	278
5.38	3mm.	279
5.39	Vowel elision before the DTP suffixes.	280
5.40	Non-permanent consonants.	281
5.41	Diphthongization.	282

List of Figures

5.42	Initial change /-ay-/.	283
5.43	Initial change /-ii-/.	284
5.44	/-yi/ loss.	285
5.45	Inverse clipping.	285
5.46	Network diagram depicting the 50 most common category sequences in the morphology of Parser 1.	288
5.47	No accented vowels.	298
5.48	Shorten.	299

List of Acronyms

API Application Programming Interface. 36, 39, 76, 94, 98, 102, 110, 123, 126–128, 151, 246, 316

CF context-free. x, 181, 182

CRUD Create, Read, Update, and Delete. 174, 176

CS context-sensitive. 63, 162, 163, 167, 168, 170–172, 246

CSV Comma-Separated Values. 85

FSA finite-state automaton. x, 165–167

FST finite-state transducer. x, 92, 159, 161–163, 167–174, 176, 177, 180–183, 186–189, 202, 204, 212, 234, 235, 245, 249, 250, 252, 254, 255, 258, 265, 269, 274, 277, 279, 285, 287, 289, 296, 297, 302, 308, 311, 313

GUI Graphical User Interface. 31, 72, 80, 88, 97, 100, 102, 123, 125, 126, 131, 140, 144, 146, 148, 151, 154, 156, 164, 203, 204, 298, 303, 315

HTML Hypertext Markup Language. 24, 28, 29, 36, 47, 89, 110, 113, 148, 149, 156, 157, 349, 355, 381, 392, 393, 397, 398

HTTP Hypertext Transfer Protocol. 36, 37, 78, 108, 110, 246

IGT interlinear glossed text. 35, 100, 207

IPA International Phonetic Alphabet. 206–209, 311

JSON JavaScript Object Notation. A standard for converting a number of data structures to and from strings. 36, 48, 75, 76, 78, 110, 174, 175, 318

List of Acronyms

- LM** language model. ix, x, 92, 159–163, 190, 192–202, 226, 227, 238, 245, 246, 248, 249, 258, 289–293, 296, 297, 299–303, 307–309
- NLTK** Natural Language Toolkit. A library of Python modules for analyzing and processing language data. 16, 78, 90
- NoSQL** Variably unpacked as *no SQL* or *not only SQL*, NoSQL refers to a type of database wherein data are not modelled as relations between tables as they are in relational databases. 70, 75, 77
- OCR** optical character recognition. 215
- OLAC** the Open Language Archives Community. 83
- SQL** Structured Query Language. A declarative language for querying relational databases. 49, 70, 75, 77, 127, 128, 134, 340
- URL** Uniform Resource Locator. 25
- UUID** Universally Unique Identifier. 132, 174, 185, 357, 358, 375, 376, 400
- W3C** World Wide Web Consortium. The primary standardization body for web technologies and specifications. 76

Acknowledgements

I am fortunate to have the opportunity to acknowledge here the many people who have supported, challenged, taught and inspired me in the pursuit of this degree. Foremost amongst these is my partner and staunchest supporter Tracy Merry, who always encouraged me to persevere, who graced me with more patience than I deserve, and who suffered as only a civilian can through the proofreading of linguistic prose.

I am especially grateful to my committee for allowing and encouraging me to pursue a non-standard dissertation topic. Henry Davis, my supervisor, has an infectious passion for linguistic theory and endangered languages fieldwork and his overtly subversive sentiments are a siren call to a malcontent such as I. In our discussions about computational and collaborative approaches to fieldwork, he was ever supportive, insightful, and laden with the knowledge of experience. Lisa Matthewson is a fearfully impressive semanticist and fieldworker, a paragon whose abilities subtly encouraged me to assume an auxiliary role in those enterprises; her thorough and discerning feedback on my work was returned with marvelous speed. I am thankful for Alexandre Bouchard-Côté's enthusiasm for my work and his guidance in aspects involving statistics, computational linguistics, and software development.

Acknowledgements

Though I have spent more time at UBC Linguistics than I sometimes care to admit, this extended tenure afforded the opportunity to learn from and be guided by two faculty members in particular. Martina Wiltschko is a dedicated and innovative teacher and prescient mentor. Without her initial encouragement I may never have begun this degree and if it were not for the vitality of the Blackfoot research group that she fostered and hosted I probably never would have been inspired to build collaborative fieldwork software. An audacious researcher and theoretician, her criticisms and challengings of my work have consistently pushed me to go further than I otherwise would have. I am also thankful to Hotze Rullmann for teaching me the basics of the only linguistic subfield that really means anything; in another life I may have trained as a semanticist under his clearheaded tutelage.

With respect to the dissertation proper, my two department-external examiners Fei Xia and Mark Turin deserve much thanks for feedback which resulted in significant improvements to its structure and clarity and to a tempering of otherwise hyperbolic claims. I am also notably appreciative of the efforts of Natalie Weber, a fellow student who went above and beyond in taking the time to read through an early draft and providing valuable feedback.

Gina Cook's in-depth discussions with me concerning software for linguistic fieldwork have had and will continue to have a profound effect on my work in this area. James Crippen and Masaru Kiyota's help in setting up web servers was critical. Notable contributors of OLD-related code, documentation, fieldwork data, and feedback are Patrick Littell, Meagan Louie,

Acknowledgements

Rose-Marie Déchaine, Michael McAuliffe, Sara Johansson, Erin Guntly, and Michael Schwan.

An exhaustive list of the many other very fine linguists, fieldworkers, and all-round stellar individuals that I have had the good fortune to know and/or learn from would of necessity include John Lyon, Heather Bliss, Solveiga Armoskaite, Amelia Reis Silva, Gunnar Ólafur Hansson, Donald Derrick, Douglas Pulleyblank, Jeff Mühlbauer, Shujun (Seok Koon) Chin, Giuseppe Carenini, Guy Carden, Jennifer Glougie, Mark Scott, and Donald Frantz.

Last but certainly not least it is a pleasure to express my most sincere gratitude to my language consultants and teachers. My generous Okanagan speakers and guides in the beautiful lands surrounding Douglas Lake—Lottie Lindley, Sarah McLeod, Nancy Saddleman, and Sharon Lindley—showed me a strength of community that I marvel at to this day. Finally, I am eternally grateful to Beatrice Bullshields for teaching me something of her language and culture, for inviting me to her home, and for introducing me to her family and friends. She taught me much about Blackfoot, self-respect, and resilience. Ever contemplative and patient, her influence on my life is probably greater than she knows.

For Orson and Phoebe.

Chapter 1

Introduction

The primary claim of this work is that the Online Linguistic Database¹ (OLD)—open source software written by the author—facilitates and expedites linguistic fieldwork. The OLD supports collaboration and the sharing and reuse of data while providing features that promote consistency and allow for powerful search, the computational implementation and testing of models of the grammar, and the automation of morphological analysis creation. Independent of whether the OLD is the tool that best implements this feature set, the present work argues that the field of linguistics stands to benefit from using such tools, from supporting and contributing to their development, and from fostering critical discussion of the collaborative and interdisciplinary methodology that they entail. The endangered status of many languages at present brings an urgency to this exhortation: the more we can do with small and declining populations of fluent speakers, the better for all parties involved. As the present work seeks to convey, many potentially rewarding opportunities and challenges await the linguists, engineers,

¹The main page of the OLD is <http://www.onlinelinguisticdatabase.org>. The source code for the OLD web service (i.e., the OLD v. 1.0) can be found at <https://github.com/jrwdunham/old> and the source for the OLD web application (i.e., the OLD v. 0.2) can be found at <https://github.com/jrwdunham/old-webapp>. See chapter 2 for an explanation of these distinctions.

computer scientists, educators, and community language activists who venture into this domain. These efforts have the potential to bring about the documentation of aspects of particular languages (and language families), the forming of linguistically interesting generalizations and analyses, and valuable contributions to efforts in endangered language revitalization.

I do not presume the OLD to be the authority or the final word in how this approach to linguistics should take form. The OLD is a contribution to the evolution of linguistic fieldwork methodology. That said, the software under discussion is a valuable tool for those interested in pursuing said approach. In addition to being practical, it exhibits interesting design decisions and features that are informed by the integrated and distilled expertise of many linguistic fieldworkers. It may be used as an inspiration or a foil for similar projects and, as it is open source and modularized, it, or parts of it, may be incorporated into such projects.

The claim that the OLD facilitates and contributes to linguistic fieldwork is supported by objective scientific evaluation in one instance (i.e., in evaluating the morphological parser creator's performance on Blackfoot), but for the most part the arguments are based on my own experience with the software (*qua* developer, fieldworker and theoretical linguist), anecdotal attestations of its value from users, feature comparison with similar tools, and *a priori* argumentation for the plausible value of its features and design. The primary original contribution to knowledge of my doctoral work, however, is the OLD itself. The main claim of this dissertation could, of course, be better supported with recourse to more objective and scientific evaluations. One evaluative approach would be to analyze user responses to

questionnaires about the software. Another would be to design controlled experiments to measure and compare the speed, accuracy, and thoroughness in the accomplishment of circumscribed fieldwork tasks of a variety of fieldwork methodologies, some including the OLD and others not. A final approach would involve real-world demonstrations of the value of the OLD’s search and modelling features to theoretical linguistic research—e.g., by writing a research paper with claims supported by data extracted via queries on OLD data sets—and the value of its data-sharing interfaces to language revitalization efforts—e.g., by creating prototypes of language learning software that make use of the OLD as a data resource.² However, since there is only so much that can be done in the finite interval that is a graduate program, I leave such extended evaluation for future work.

The dissertation is structured as follows. Chapter 2 argues for the effectiveness of the OLD as a tool for linguistic fieldwork by describing and evaluating its most valuable and innovative features and by comparing it to similar tools. Chapter 3 describes the morphological parser creator, an application added to the core feature set of the OLD. Chapter 4 briefly describes the Blackfoot language and details a data set of that language created using the OLD. Chapter 5 describes and evaluates the performance of two morphological parsers for Blackfoot that were built using the parser creator and data set described in the preceding two chapters. Finally, Chapter 6 concludes with a summary and a brief discussion of potential future improvements to the software. The remainder of this introductory chapter

²See Bird et al. (2013) for an example of using this evaluative approach on a specific set of language documentation methodologies and technologies.

fleshes out the argument for the OLD.

1.1 The argument

Understanding a piece of software requires understanding the problem it is trying to solve. The problem addressed here is how linguistic fieldwork might be better accomplished.

Many of the world’s languages are in danger of losing their last remaining fluent speakers and thus becoming “sleeping languages” (Leonard, 2008).³ It is widely predicted that by the end of the century more than half of the world’s 7,000 languages will no longer be spoken (cf. Harrison, 2007). More than twenty years ago, Hale et al. (1992) estimated that 80% of the then-spoken Native North American languages were moribund. More recently, one source (First Peoples’ Heritage, Language and Culture Council, 2010, p. 22) asserts that all 32 of British Columbia’s “First Nations languages are critically endangered, if not sleeping already.”

If one accepts that endangered languages are valuable, that their documentation, analysis, and perpetuation are goals worth pursuing, then a next logical question is how such goals might be better accomplished. This is a big question with a multifaceted answer.⁴ However, one aspect of linguistic fieldwork that, in my opinion, shows ample room for improvement is the development of software that facilitates collaboration, data-sharing, and the

³Leonard (2008) uses the term *sleeping*—in contrast to *extinct*—to emphasize the possibility that, through community-lead revitalization efforts, currently unspoken languages may one day be spoken again.

⁴For an argument for the value of a theoretically-informed hypothesis-driven approach to the study of linguistic diversity see Davis et al. (2014).

automation and acceleration of menial tasks and, to the extent possible, of higher-level tasks such as the testing of hypotheses.

Assessing exactly which features are most crucial to such software requires an understanding of the workflow of linguistic fieldwork. While the nature of this workflow is explored below, a preview of the requirements is given in the following list of exhortations. It should be easy to find relevant data quickly; data should not be unnecessarily re-elicited; fieldworkers with a diverse range of goals should be able to access and make use of one another's data; and features should exist which facilitate the computational implementation of analytical models such that both the empirical testing of and the generation of analytical representations based on such models can be automated. The OLD seeks to meet these requirements and this dissertation argues that it succeeds. In what follows, I motivate and elaborate on the sub-claims of this argument.

1.1.1 **Endangered languages are valuable**

Languages—even and especially those with very few fluent speakers—are valuable from a variety of perspectives and efforts that seek to document, analyze, and perpetuate them should be supported. This statement has political significance and is not uncontentious. Hale et al. (1992) argues that linguists working on endangered languages have a duty to contribute to their perpetuation and revitalization. Ladefoged (1992) responds that the exhortation of Hale et al. (1992) is too general since some communities would rather pursue linguistic homogeneity over minority language vitality for, say, purposes of national cohesion. Dorian (1993) responds by asserting that

1.1. *The argument*

the scenarios discussed by Ladefoged (1992) are exceptional and that, even in these rare cases, subsequent generations tend, as a rule, to regret the loss of their minority languages. In my experience, communities do desire to see their languages better documented and more widely spoken and they are actively engaged in efforts to achieve these ends;⁵ in these cases, academic linguists do have both an ethical obligation and a self-interested motivation to reciprocate by supporting these endeavours.

It should come as no surprise that communities value their endangered and sleeping languages as vessels of culture; however, sometimes unconsidered are the scientific and more broadly sociological values of language. From the theoretical linguistic point of view, thorough study of the widest possible range of human languages is crucial to understanding human cognition and social anthropology. Equally if not more important is the value of a language to the social well-being of its community and, as a result, to the larger social groups within which that language community is embedded. In the North American context, this means that aside from any reparations owed for the language and culture-eroding practises so infamously epitomized by the residential/boarding schools (Churchill, 2004), the dominant cultures should recognize a selfish interest in fostering a healthier society by supporting the flourishing of endangered languages.

Endangered languages are scientifically valuable. Theoretical linguistics

⁵Examples of community-led language revitalization and documentation efforts abound. Two examples are the Breath of Life biennial language restoration workshop (sponsored by the Advocates for Indigenous California Language Survival, the Department of Linguistics at the University of California at Berkeley, and the Townsend Center) and the community-based revitalization certificate program and N'syilxcen language classes offered by the En'owkin Centre of British Columbia, Canada.

is centrally concerned with exploring the possibility space of natural language, i.e., the extent to which languages can vary and the properties that are universal to all. Unsurprisingly, linguistic analyses since the rise of the generative tradition in the 1950s have been supported primarily by data from majority languages, especially the western European ones spoken by most linguists. However, the past several decades have witnessed a resurgence⁶ in interest in greater empirical coverage. In the generative linguistic tradition, it has long been argued that linguistic competence is too complex to be plausibly learnable via general cognitive mechanisms, given the relatively impoverished data available during acquisition (Chomsky, 1980). It is therefore assumed that a significant portion of human linguistic knowledge must be innate. The dominant research program in this tradition thus consists in rigorously describing and analyzing the properties of the greatest possible subset of natural languages with a mind to discovering universal principles and well-defined parameters of variation.⁷

Other traditions within theoretical linguistics and other fields related to linguistics are also interested in the analysis of endangered languages. Historical linguistics, for example, seeks to understand the relatedness of extant languages and to reconstruct proto-language ancestors. Anthropological linguistics has long been interested in the analysis of endangered languages and

⁶Of course, prior to the rise of the generative tradition, much work was done on minority languages by structuralist linguists such as Leonard Bloomfield.

⁷Note that there is an ongoing debate over what is the best approach to studying linguistic diversity. Davis et al. (2014) argues the merits of a theoretically-informed, hypothesis-driven generative/universalist approach. Levinson and Evans (2010) (among others) argue for large-scale typological analyses of descriptive works. However, this debate is orthogonal to an argument for the OLD since this tool should be useful to adherents of either point of view.

1.1. *The argument*

their interrelation with culture. Clearly, endangered languages are crucial objects of study for those seeking to understand human cognition, culture, and history. The common denominator here is an interest in endangered languages for their value in relation to our knowledge of humanity.

Communities whose ancestral languages are endangered are also very interested in their documentation and revitalization, though for a somewhat different set of reasons. Their languages encode ways of thinking, traditional knowledge, and narratives that are vital to their cultures, identities, and health. With respect to the last point, at least one study has shown a significant positive correlation between ancestral language use and metrics of societal health: Hallett et al. (2007, p. 398) finds a correlation between higher levels of Aboriginal language use in British Columbia First Nations bands and lower youth suicide rates and concludes that “indigenous language use, as a marker of cultural persistence, is a strong predictor of health and wellbeing in Canada’s Aboriginal communities.” Within endangered language communities there are many motivated, hard-working, and skilled individuals who are contributing a great deal to these efforts. Unfortunately, communities with endangered languages tend also to be politically marginalized, economically disadvantaged, and small and, as a result, the resources for fully successful documentation and revitalization are often not available.

Because endangered languages are valuable scientifically and to the health and well-being of their communities, they should also be considered valuable to society at large. That is, in purely practical terms, the spiritual/cultural health of a community—directly tied to language—means its individual members are healthier, and therefore use considerably fewer resources of the

larger societies in which they are embedded. The more we can do to document, analyze, and revitalize these languages, the more we can learn about human cognition, culture, and history. Perhaps more importantly, such efforts can help us to begin to redress the legitimate grievances of indigenous communities for the unethical language- and culture-eroding practises inflicted by our governments and religious institutions. Beyond recognition and atonement, success in the documentation and revitalization of endangered languages has the potential to strengthen disadvantaged subgroups within our societies and thereby benefit us all.

Assuming, then, that the study of endangered languages is worthy of support, I would at this point like to discuss the nature and practise of linguistic fieldwork so that the reader may come to an understanding of the process by which speaker knowledge is transformed into linguistic artifact.

1.1.2 **Fieldwork**

Fieldwork, in one sense, is the act of transforming linguistic knowledge into real-world artifact. Different scenes within this act are labelled according to the complexity and nature of the artifacts generated. First-order documentation is a collaboration between researcher and language consultant which produces primary data types like transcriptions, translations, annotations, basic metadata, and audio recordings as well as morphophonological analyses. From these primary artifacts, second-order documentation constructs higher-level data types such as dictionaries, grammars and representations of narratives. The products of documentation are used by researchers for the creation of academic publications and, ultimately, the accumulation and

refinement of scientific knowledge (cf. Woodbury, 2003). All of these linguistic data types are used in the creation of artifacts relevant to the task of revitalization, such as language textbooks, exercises, and software.

The view of fieldwork as the reification, transportation, and transformation of linguistic data highlights the interrelatedness of its auxiliary endeavours. Under this view, successful revitalization is the production of language data in its richest form, viz. as grammatical knowledge in the minds of future generations of speakers. In the context of endangered and moribund languages, therefore, revitalization is both a primary goal of documentation efforts and a requirement for the perpetuation of research efforts.

Fieldwork is carried out by academics, language communities, and passionate individuals and groups. As a member of a linguistics department in a research-oriented university, I am most knowledgeable of the motivations and workflow of academic linguistic fieldworkers. However, because of the interconnectedness just elaborated, I am also familiar with the documentary and revitalization-related domains. This imbalance in perspective is a natural outcome of the division of labour necessitated by a complex endeavour like linguistic fieldwork. More importantly, it is directly relevant to the aim of this dissertation, that being to demonstrate methodologies and technologies which increase the availability and usefulness of researcher-generated artifacts to fellow fieldworkers.

Theoretical linguists seek to increase scientific knowledge of linguistic competence. The data they elicit (i.e., gather) from speakers are designed to answer research questions relevant to the verification and falsification of linguistic theories. However, getting to a stage where such questions might

even begin to be asked requires at least a basic level of proficiency with the language under study. Linguists interested in complex syntactic, semantic, and pragmatic phenomena are likely to require an even higher level of proficiency. In addition, linguists often have a holistic appreciation for the languages they study, often including a motivation to give back to the speakers and communities that make their work possible. From the opposite perspective, communities do not want to be the objects of academic research unless academics reciprocate by donating their skills and resources to revitalization. For these reasons, linguists end up eliciting foundational linguistic forms and also contribute to the production of materials that are relevant to a broader audience, e.g., dictionaries (Lyon and Greene-Wood, 2007), learning grammars (Davis, 2012), collections of translated and analyzed texts (Matthewson, 2005; Lindley and Lyon, 2012), and expositions of theoretical analyses in terms appropriate to non-linguist learners (Davis, 2012).

Linguistic data are elicited in a variety of ways. The technique with the longest history, and arguably that which is least artificial, involves recording and transcribing the speech of one or more speakers, as when narratives and conversations are gathered to produce texts (Boas, 1917). A common practise is to ask for translations of metalanguage (e.g., English) forms, sometimes supplying relevant contextual information via verbal description, images and/or videos. Some fieldworkers seek to sidestep unwanted intrusions of the metalanguage by constructing stimuli (e.g., images or videos) that elicit a response in the speaker without providing a metalanguage form for translation (Yegerlehner, 1955). Another technique with similar moti-

vations involves describing contexts and asking questions using the object language itself, assuming the researcher is proficient enough to do so. However, see Matthewson (2004) and AnderBois and Henderson (to appear) for nuanced discussions of the use of the metalanguage in translation-based elicitation and a defence of that practice in certain cases. Finally, a fruitful elicitation method—which is contentious to some, cf. Mithun (2001); Dimmendaal (2001)—involves requesting speaker judgments on forms produced by the researcher, thus making it possible to gather data that specify ungrammatical, questionable, or contextually infelicitous forms. For a discussion of elicitation techniques in linguistic fieldwork generally see Bower (2008) and Newman and Ratliff (2001), and for semantic fieldwork in particular see Matthewson (2004), Krifka (2011), and Bochnak and Matthewson (to appear).

The foundational products of linguistic fieldwork are representations of linguistic forms at a variety of levels, from morphemes to words, phrases, sentences, and multi-sentential objects such as conversations and narratives. The core components of such representations are transcriptions; these may be written in an orthography specific to the language or in a more general-purpose system, e.g., the International Phonetic Alphabet (IPA) (International Phonetic Association, 1999) or the Americanist Phonetic Alphabet (APA) (Goddard, 1996); they may be phonetic, phonemic, or some hybrid. In addition to transcriptions, also generated are translations, indications of grammaticality, comments and questions of the researcher, comments of the speaker(s), and metadata specifying when, where, how, and with whom the data were elicited. If stimuli are used to build context or otherwise con-

strain experimental variables, these should also be considered products of linguistic fieldwork. Audio-video recordings are a data type that has a lot of potential for reuse, especially if transcriptions and recordings of utterances are time-aligned.

A widespread and increasingly standardized (cf. Bickel et al., 2008) practise is the morphological analysis of linguistic forms and the generation of a particular type of multi-line representation of such analyses known as *inter-linear glossed text* (IGT). This typically involves, minimally, a transcription line indicating morpheme boundaries, followed beneath by a line of glosses corresponding to the morphemes, and terminated by a list of one or more translations. Though there are variations on this pattern, linguists from a broad spectrum of traditions make use of this type of representation.

Other types of analyses generated in the course of fieldwork may include syntactic category tags, phrase structure representations, semantic representations, and annotation data structures, e.g., Praat TextGrids and ELAN files.

Building upon such *foundational* artifacts, linguists of various stripes go on to generate one or more of the following: descriptive grammars, dictionaries, publication-worthy representations of narratives and stories, pedagogical tools, corpora, treebanks, databases, and research papers.

1.1.3 **Better fieldwork**

Given an understanding of fieldwork and its importance from a range of perspectives, this dissertation asks *how can we do fieldwork better*. The answer proposed is *share data and automate tasks*.

1.1. *The argument*

As we have just seen, linguistic fieldworkers generate a lot of primary data as well as higher-level artifacts that are potentially useful to their fellow fieldworkers. In practise, however, a significant portion of primary data elicited are stored away in private collections of handwritten notes or digital documents and are never reused or even reviewed. Even higher-level artifacts such as dictionaries may be underexploited because the data they contain are not structured for accessibility.

In order to move from potentially useful to maximally reused, we need to make the sharing of data both easy and desirable. First of all, this means building up the basic infrastructure for transferring data from researcher to researcher, while structuring and delivering the data in a way that facilitates reuse. A system that encourages the sharing of data will have higher adoption rates if individual fieldworkers can see how it will immediately benefit their own work. In addition, therefore, to the enticement of accessing the data of other researchers, a data-sharing system should provide further conveniences, the automation of fieldwork-related tasks being a primary example of such.

If all fieldworkers could quickly, accurately, and ethically gain access to relevant data in their own collections and those of their peers, then theoretical questions could be answered, documentary artifacts could be generated, and learners could become fluent speakers more quickly. Another important benefit arising from greater transparency and access with respect to primary linguistic data is that theoretical claims based on such data could be more readily and rigorously scrutinized and criticized, thus leading to more robust analyses and greater confidence in theoretical claims grounded in data

from minority languages.⁸ Depending on the nature of the analysis (e.g., statistical corpus-based), the ready availability of the data may also allow for verification via experimental reproduction.

1.1.4 The Online Linguistic Database

The OLD is software designed to help fieldworkers do fieldwork better. The OLD is used to create language-specific web applications that facilitate the collaborative creation and curation of databases of fieldwork data. An OLD application is at its core a web interface to a database which has a structure (i.e., schema) designed for linguistic fieldwork and which can be altered and viewed by multiple users concurrently. This core feature—i.e., the multi-user, web-based, fieldwork-oriented database-ness—is what makes collaboration and data-sharing possible: contributors create representations of linguistic forms and related data types and other users can immediately use these data to inform their own research, documentation, and revitalization projects.

Other salient features of the OLD contribute to achieving the goal of more efficient use of linguistic fieldwork data. Advanced search facilitates fast and targeted data retrieval. Features encouraging consistency and standardization of both analysis and representation make data easier to understand and reuse. Model-implementing features contribute to the testing of the analyses and their underlying frameworks and assumptions, as well as to

⁸This is not to say that researchers whose work is grounded in endangered language fieldwork data are less trustworthy. It is simply an acknowledgement that it is very difficult for a researcher to independently re-elicite the endangered language data which is asserted, by a fellow researcher, to provide evidence for a given theoretical claim. If the data were more accessible, critics might be better able to discover cases where an analysis is inadequately supported by them.

the automation of analytical tasks, including the generation of further representations (e.g., morphological analyses), the existence of which in turn improves targeted search.

Some implementation details are crucial enough to achieving the goal of more efficient fieldwork that they could be considered core features. The OLD is open source software written primarily in Python, a programming language that is designed with readability in mind and which, because of its extensive string manipulation constructs and because of the existence of the Natural Language Toolkit (NLTK) (Bird et al., 2009), a library written in the language, is well-suited to linguistic and language-processing applications. These properties allow for users to contribute to the code, use parts of it in their own projects, and/or read it for feature implementation ideas.

As of the current development version (i.e., 1.0), an OLD application is also a *web service*, which means that it exposes a standardized interface for programmatic interaction. This makes the data stored easy to access for a variety of purposes. An OLD web service can be used as a data-serving module within a larger application or service. For example, an OLD web service could be used in multi-language applications for cross-linguistic and typological analysis as well as revitalization-relevant applications such as audio dictionaries and learning tools. More mundanely, the data stored in a language-specific OLD web service can be downloaded to a contributor's local system and processed as necessary. Chapter 3 illustrates this by showing how morphologically analyzed words can be extracted from an OLD web service in order to evaluate the performance of computationally implemented morphophonological models.

The OLD has been in existence (in one form or another) for six years. There are currently nine language-specific OLD web applications targeting Blackfoot (Algonquian), Coeur d'Alene (Salish), Gitksan (Tsimshianic), Ktunaxa (isolate), Kwak'wala (Wakashan), Nata (Bantu), Okanagan (Salish), Plains Cree (Algonquian) and Tlingit (Na-Dené). Data in these applications have been contributed and used by a geographically and motivationally diverse group of fieldworkers, including four field methods classes offered by the UBC Department of Linguistics. The process of adapting to these diverse languages and fieldworker types has helped to improve and mature the software.

1.2 Summary

Linguistic fieldwork, broadly construed, is the generation of language artifacts and is valuable scientifically, culturally, and socially. Because of their interrelatedness, distinct sub-endeavours contributing to the fieldwork enterprise (i.e., documentation, research, and revitalization) stand to benefit from cooperation. There are, therefore, exciting opportunities for methodological improvement in this domain. The Online Linguistic Database illustrates a particular approach to improving fieldwork methodology which involves the development of software that facilitates collaboration and data-sharing, as well as the provision of conveniences such as task automation and computational analysis implementation.

This dissertation describes and evaluates the OLD as a tool for improving fieldwork methodology. Chapter 2 introduces and describes the software,

1.2. Summary

compares it to similar tools, justifies its design decisions, and argues that it is an effective fieldwork tool. Chapter 3 describes the morphological parser creator, an application added to the core functionality of the OLD, which can be used to build morphological parsers to expedite data entry and test theoretical models. Chapter 4 describes the Blackfoot language and a data set for that language created using the OLD. Chapter 5 describes and evaluates two parsers built using the morphological parser creator and the Blackfoot data set. Finally, chapter 6 summarizes the argument for the OLD and discusses some exciting possible developments to the software.

Chapter 2

Fieldwork

This chapter describes the Online Linguistic Database (OLD) and argues that it is a valuable fieldwork tool. Its primary goal is to facilitate collaborative language documentation and the sharing and reuse of fieldwork data. The salient secondary features discussed here contribute to the primary data-sharing goal by facilitating the creation of consistently structured, easily re-purposable, effectively searchable, and well presented data. The descriptions of these features and the demonstrations of their utility to fieldwork constitute the argument for the software.

The OLD is open source software for creating web applications designed to facilitate collaborative linguistic fieldwork. A language-specific OLD application is a web-accessible repository of linguistic artifacts contributed by multiple fieldworkers. The system provides data structures, representations, interfaces, and conveniences that make it easy for diverse researchers to find and use the data they need.

Linguistic fieldwork is the transformation of speaker knowledge into language artifacts. Language documentation, research and revitalization are all endeavours that involve fieldwork. Documentation seeks to describe, record and preserve a language. Research seeks to analyze linguistic data and arrive

at insightful generalizations and explanations. Revitalization seeks the perpetuation of waning languages in the minds of future generations of speakers, and involves the creation of artifacts relevant to language acquisition and florescence. All of these endeavours involve elicitation, i.e., the generation by fieldworkers and speakers of primary language artifacts such as transcriptions and recordings.

Moreover, all of these endeavours are interrelated such that successes in any one facilitate successes in all of the others. For instance, a descriptive grammar produced as part of a documentation effort invariably contains data and generalizations that will constitute the groundwork for the formulation of research questions and the production of revitalization-specific artifacts. Conversely, descriptive grammars are never truly theory-neutral but are informed by the assumptions and theoretical linguistic research of some tradition; in addition, writers of descriptive grammars often cite research papers from outside of their tradition for generalizations or data points.⁹

The OLD is founded upon the premise that the primary language data generated in all of the diverse types of fieldwork are ripe for reuse. A researcher investigating, say, the syntax of nominal expressions will benefit from easy access to peer-generated data. Of course, from a pure research perspective, the value of peer data decreases as differences in analytical framework and research focus increase. However, the reader of a treatise on the methodology of linguistic fieldwork can be assumed to hold to a wider perspective, to be involved in projects for the advancement of documentation

⁹I am sympathetic to the arguments of those (cf. Murray, 2014) who argue that theory-driven research questions can uncover data points and insights that would not otherwise be discovered and which are valuable to documentation and revitalization goals.

and revitalization, and to find value in data elicited by fellow fieldworkers with a diverse range of goals. Even without this proviso, it is clear that, at the very least, research projects in their initial stages will find value in diversely motivated and authored fieldwork data.

This chapter is divided into three sections. Section 2.1, *Overview of the OLD*, provides a high-level summary of the software, including a succinct description, a history, and a review of the implementation. Section 2.2, *Other fieldwork tools*, reviews Toolbox, FLEx, and LingSync and compares these to the OLD. Section 2.3, *Features*, describes the core features of the OLD and argues for their value to the fieldwork enterprise. Section 2.4 summarizes.

2.1 Overview of the OLD

This section provides an overview of what the OLD is and how it works. It begins with a short history of the circumstances that gave rise to the software and the events and considerations that influenced its development. From there it moves on to a mid-level description of the software in its entirety and ends with some technical implementation details.

2.1.1 History

The OLD, in one form or another, has been continuously in production and under development for the past six years. It arose from my own experiences doing fieldwork on Blackfoot. The current production¹⁰ version (0.2) has nine language-specific web application instantiations. The current develop-

¹⁰Software developers use the jargon of *development* and *production* versions of software to refer to versions that are under active construction and in use, respectively.

ment version (1.0) introduces the morphophonological modelling functionality and the architectural restructuring that makes an OLD application a web service. Once a user interface for the development version is completed, currently active applications will be migrated to it.

I began my doctoral studies with the intention of pursuing formal semantic analyses of aspectual phenomena in understudied languages. In particular, I was interesting in understanding the form, distribution, and meaning of the temporal functional morphology of Okanagan. The department of Linguistics at the University of British Columbia is well-known for its strength in theoretical linguistic research grounded in understudied and endangered language data and, as such, is an excellent location to pursue such a proposed course of work.

However, I soon became frustrated with various obstacles that discourage timely access to relevant data. Linguistic fieldwork is difficult and data are hard-won. Often one realizes that evidence pertinent to a particular theoretical claim has already been elicited. Yet scouring one's handwritten notes or digital documents is slow going and not guaranteed to pay off. Even more difficult to retrieve are the relevant forms that colleagues mention having elicited. Re-elicitation is often the ultimate course of action, an inefficiency whose undesirability can readily be grasped when one recalls that the languages under study are in danger of extinction.¹¹

While working with groups of fieldworkers in two full-year linguistic field

¹¹This is not to say that re-elicitation is necessarily an undesirable inefficiency. In many cases it can be extremely useful to re-elicite data in order to ensure their replicability and the robustness of the generalizations that they evince. That said, in other cases re-elicitation is clearly undesirable, redundant, and inefficient.

2.1. Overview of the OLD

methods courses (on Blackfoot and St’át’imcets) and with an independent group on the Okanagan language, I came to believe that the data elicited by such groups could, if consolidated, consistently structured, and rendered accessible, prove highly useful to documentation, research, and revitalization efforts.

The precursor to the OLD was the Blackfoot Language Database (BLD) (Dunham, 2013b), a web application designed to facilitate collaborative fieldwork for myself and a group of Blackfoot researchers which had emerged from a field methods course on that language. As my fieldwork experience broadened into other languages, I generalized the BLD to the Online Linguistic Database (OLD), language-non-specific collaborative fieldwork software.

The current production version of the OLD is 0.2. A demo application is currently being served at <http://www.onlinelinguisticdatabase.org> and the source code is available on GitHub.¹² Among the nine language-specific OLD 0.2 applications currently in use¹³ are applications for two languages that I have research and fieldwork experience with, namely Blackfoot¹⁴ and Okanagan.¹⁵

In using these applications for my dissertation research, I became increasingly interested in potential improvements to the software, of which two—consolidation of the server-side logic into a web service and the provisioning of morphophonological modelling capabilities—necessitated a rewrite. The

¹²GitHub is a “web-based hosting service for software development projects that use the Git revision control system” (GitHub, 2014). Git is software that helps groups of developers to collaboratively write complex software in such a way that modifications from multiple contributors can be tracked and handled logically.

¹³See § 2.1.3 for an explanation of what is meant by *language-specific OLD application*.

¹⁴<http://bla.onlinelinguisticdatabase.org>.

¹⁵<http://oka.onlinelinguisticdatabase.org>.

2.1. Overview of the OLD

source code for this rewrite, i.e. the OLD 1.0., can be found on GitHub.¹⁶

Documentation detailing the data structure and installation methods has been written for the OLD 1.0 and is available in Hypertext Markup Language (HTML) and PDF formats.¹⁷ The OLD 0.2-based applications have user-oriented documentation embedded within their interfaces.¹⁸

Within the language-specific OLD applications currently in use, there are about 19,000 *forms* (primarily sentences), 300 texts, and 20 GB worth of audio files. There are 180 registered users across all applications, of which 98 have entered and 87 have elicited at least one *form*. The applications for Blackfoot, Nata, Gitksan, Okanagan, and Tlingit are seeing the most use. The exact figures are given in Table 2.1.

language	<i>forms</i>	texts	audio	GB	speakers
Blackfoot (<i>bla</i>)	8,847	171	2,057	3.8	3,350
Nata (<i>ntk</i>)	3,219	32	0	0	36,000
Gitksan (<i>git</i>)	2,174	6	36	3.5	930
Okanagan (<i>oka</i>)	1,798	39	87	0.3	770
Tlingit (<i>tli</i>)	1,521	32	107	12	630
Plains Cree (<i>crk</i>)	686	10	0	0	260
Ktunaxa (<i>kut</i>)	467	33	112	0.2	106
Coeur d'Alene (<i>crd</i>)	377	0	199	0.0	2
Kwak'wala (<i>kwk</i>)	98	1	1	0.0	585
TOTAL	19,187	324	2,599	19.8	

Table 2.1: Data in OLD applications (Feb 14, 2014)

Note that the values in the speakers column of Table 2.1 are taken from

¹⁶The OLD is also available on the Python Package Index and is installable via both the EasyInstall and Pip Python package managers. See <https://pypi.python.org/pypi/onlinelinguisticdatabase/0.2.9> and <https://pypi.python.org/pypi/onlinelinguisticdatabase/1.0a1>.

¹⁷<https://online-linguistic-database.readthedocs.org>.

¹⁸<http://www.onlinelinguisticdatabase.org/help/olduserguide>.

2.1. Overview of the OLD

Ethnologue¹⁹ and are provided only to give a rough indication of the speaker populations of the languages. The parenthesized three-character strings in the first column are the ISO 639-3²⁰ identifiers of the languages. The Uniform Resource Locator (URL) for each language-specific OLD application is the ISO 639-3 identifier followed by `onlinelinguisticdatabase.org`.

The OLD has evolved in response to usage by these groups and owes much to their feedback and suggestions.

2.1.2 Strategy

The purpose of the OLD is to make linguistic fieldwork easier, more efficient, and more rewarding. I contend that sound strategy in achieving this involves adapting to extant fieldwork practises and supplementing them via a) collaboration facilitation and b) the provision of improvements to low and mid-level task completion, thereby freeing up more fieldworker time for high-level creative work.

Adapting the software to current practises is more than just an olive branch to methodologically conservative fieldworkers. It falls out from the fact that the OLD is not a manifesto for radical methodological change at the individual level. On the contrary, the system is built on the premise of continued additive and complementary change at the communal level. I see this as a broadening in the sphere of fieldwork tasks and a consequent division of labours amongst well-rounded fieldworkers alongside software developers, transcribers, parsers, computational linguists, armchair linguistic

¹⁹<http://www.ethnologue.com>

²⁰<http://www-01.sil.org/iso639-3>

researchers, pure theoreticians, and all desired permutations thereof.

Of course, I do not presume that the OLD can, or should, transform elicitation, transcription, and translation into easy tasks. The OLD's primary aspiration is precisely to free up more fieldworker time for this difficult and time-consuming yet rewarding work.

2.1.3 Description

The OLD is open source software for creating web applications that facilitate collaborative linguistic fieldwork. Contributors enter their fieldwork data and are able to search, browse, export, and create documents from those entered by themselves and their counterparts.

The core data type defined by the OLD, and the one most commonly manipulated by contributors, is the linguistic *form*. A form is a representation of a component of a language that is at least a morpheme and at most a sentence. Linguists will recognize this seemingly arbitrarily demarcated unit as the ubiquitous datum of most research papers. Distinctions between morphemes, words, phrases, and sentences are made either implicitly—e.g., via the presence of morpheme and word delimiters in transcriptions—or explicitly via user-specified syntactic categories. Abstractly, a form is an object with properties (a.k.a. attributes) whose values are variously optional or mandatory, singular or plural, free-form text or forced choice.

Minimally, a form has values for a transcription property and for at least one translation property. Multiple transcription types may be supplied, viz. orthographic, narrow phonetic, and broad phonetic. Multiple translation values are also possible, as is a morphological analysis speci-

2.1. Overview of the OLD

fied via the assignment of values to the two attributes morpheme break—a sequence of phonemes segmented into words and morphemes by whitespace and delimiters—and morpheme gloss—a sequence of corresponding glosses segmented just like the morphemes in the morpheme break value. Secondary contentful form attributes include grammaticality (including felicity in a context), appropriateness of translations, general comments, comments from the speaker, syntactic category, and user-defined tags. Information about data provenance can also be specified via values for textual source, speaker (consultant), elicitor, elicitation method, and date elicited.

Certain form properties are assigned values automatically by the system. These include a unique integer identifier, the entry timestamp, and the enterer. If morphemes are recognized in the values supplied for the morpheme break and gloss properties, the system will also auto-generate category string and break-gloss-category values. The former is a string of morpheme categories implicit in the user-supplied morphological analysis. The latter is an interleaving serialization of the tripartite (i.e., break, gloss, category) morphological analysis data. Thus a form for French *chiens* ‘dogs’ might have *N-Num* as its category string value and *chien|dog|N-s|plural|Num* as its break-gloss-category value. These auto-generated analytic values are highly useful in improving search; details of their representation, generation and utility are given in § 2.3.8.

The OLD interface strives to make form entry and update quick and pleasant by providing conveniences such as keyboard shortcuts, tab-based navigation, and strategic auto-retention of previously entered values.

After creation or modification, a form is displayed in highly readable

2.1. Overview of the OLD

interlinear glossed text (IGT) format, i.e., with transcriptions, break, gloss, and translations in rows and their words aligned into columns. Within a morphologically complex form, morphemes specified via the break and gloss values are displayed as HTML links to matching lexical entries already in the database. To illustrate, an OLD application for French containing a form for the noun *chien* ‘dog’ and another for the number agreement suffix *s* ‘plural’ would, upon creation of a *chiens* form with *chien-s* and *dog-plural* as break and gloss values, respectively, display these component morphemes as links to the corresponding mono-morphemic form objects. This furnishes immediate feedback about whether (and the extent to which) the morphological analyses provided are consistent with the lexical items already in the system. This seemingly minor feature actually turns out to be quite effective at promoting (both inter- and intra-researcher) consistency in morphological analysis and is thus indirectly a boon to search.

OLD forms may also be associated to one or more representations of media files. These *file* objects are a data type built upon digital files such as audio or video recordings, images, and textual documents (e.g., PDFs). An illustration of the practical application of many-to-many form-file association would be a form representing a sentence associated to a video recording of the entire elicitation session during which it was gathered, an audio file of the speaker uttering just the sentence, an audio file of the elicitor uttering the translation, an image file used to provide context during elicitation, and a PDF of a research paper containing claims to which the sentence is relevant.

The digital file resource that constitutes the only mandatory value of

2.1. Overview of the OLD

an OLD file may be hosted within the OLD application or on an external site (e.g., YouTube, Vimeo, etc.). Additional file attributes may be given values, including user-created general-purpose tags, a description string, and provenance metadata.

Forms and files can be used to construct instances of the third primary data type of an OLD application: collections. Collections, at their most basic, are ordered lists of forms and, at their most complex, are multimedia documents such as research papers with embedded media files. The core attribute of a collection is its contents, a string of text which may contain references to forms. These references adhere to a simple syntax that makes use of the unique integer identifier value of a form in order to embed a representation of the form within the collection. The text may, in addition to the form-embedding references, contain prose and this may be given formatting (via a lightweight markup language). Finally, the collection as a whole may be associated to multiple file objects. When compiled, a collection is displayed as HTML and the end result is a document with formatted prose (e.g., boldface, italics, itemized and enumerated lists, section headers, tables, etc.), enumerated linguistic examples (i.e., forms) in IGT format, and embedded media files. Collections may thus be used to create a range of documents, e.g., elicitation records, representations of narratives, lesson plans, and research papers. These can be exported as LaTeX source files and compiled to PDF. Given the technologies used, export to Office Open XML (i.e., .docx) format is also possible, though this has not yet been implemented.

The last core object is the corpus. An OLD corpus is (like a collection) fundamentally an ordered list of form objects, and thus a sequence of mor-

phemes, words, and/or sentences. Multiple corpora may be created within a single OLD application. A single-file representation of a corpus is created in a process dubbed *generation*, which involves gathering specified values of the corpus' forms and writing these to disk according to a specified format.

Currently, the only corpus generation format is *trebank*,²¹ which writes to file the corpus' forms' non-empty *syntax* values, by convention phrase structure representations in Penn Treebank-compatible bracket notation. A generated OLD trebank corpus can be compiled to a binary format that can be searched structurally via the TGrep2 utility. The OLD provides an interface to the TGrep2 search function so that trebank corpora can be searched for structural patterns. This means adding to search the capacity to return only forms with, say, an indirect and a direct object.

In addition to facilitating structural and cross-sentential search, corpora can form the foundation of certain morphological components that are needed for the morphophonological modelling and parser functionalities (see below). Lexica can be extracted and morphotactic rules induced from them. Corpora can also supply the *N*-gram counts needed to build the morpheme language models that are essential to the disambiguators needed for functioning parsers (cf. chapter 3).²² Since a corpus can contain (i.e., reference) any number of forms any number of times, language models (and

²¹A simple next corpus generation format would be a string of orthographic transcriptions, in which cross-sentential patterns could be searched. A particularly useful additional generation method would be the construction of an NLTK-compatible corpus file; this would permit an OLD interface to the NLTK corpus interface and the search conveniences that it provides.

²²I use the term *disambiguator* here to refer to a function that takes a set of possible parses as input and returns them as a list sorted according to their probability. Since the most probable parse is the one that the parser will suggest to the user, the net effect is to *disambiguate* the parse set.

thence disambiguators) can be strategically biased via corpus design.

This segues nicely into the morphophonological (i.e., morphological and phonological) modelling and parser functionality provided by the OLD and discussed in detail in chapter 3. In the OLD, a morphology is a mapping between sequences of phonemes and sequences of morphemes; an OLD phonology is a mapping between phonetic/orthographic representations and sequences of phonemes. The lexicon and morphotactic rules that constitute a morphology are extracted by the system from corpora specified by users. A phonology is specified by the user as an ordered list of rewrite rules. Both morphologies and phonologies are implemented as finite-state transducers. A fully assembled OLD morphophonology is a function that parses transcriptions into sequences of morphemes and, conversely, generates surface transcriptions from morpheme sequences. Multiple morphologies, phonologies, and morphophonologies can be created within a single OLD application.

Since these components encode licit mappings, they could be used to provide feedback to users indicating whether their representations are consistent with a given analysis. For example, a user-specified phonetic transcription may not be consistent with a user-specified morphophonemic segmentation according to a certain phonological analysis. The interface could alert the user to this fact,²³ thus facilitating error correction and, more interestingly, analysis modification.

Beyond providing feedback on representational consistency, morphopho-

²³As there is, as yet, no Graphical User Interface (GUI) for the OLD v. 1.0 web service, the morphological parsers and their components are not currently being used to alert users to inconsistencies between their morphological analyses for particular words and the grammatical models encoded in their parsers. Future work on the software will involve creating a GUI which does precisely this.

2.1. Overview of the OLD

nological components can be combined with statistical disambiguators to form parsers that can automate the creation of morphological analyses. That is, given a transcription specified by a user, an OLD application can use a specified parser to automatically assign values to the form's morpheme break, morpheme gloss, and category string attributes. In principle (though this is not yet implemented), the reverse operation could also be implemented using the same components (i.e., morphophonology and disambiguator), thus allowing users to specify morphological analyses and have the system auto-generate the transcription.

In addition to creating and modifying forms, files, collections, corpora, morphologies, phonologies, disambiguators, and morphological parsers, OLD contributors create and modify a host of other (relatively) minor objects, including sources (using the BibTeX data structure), speakers, general-purpose tags, (morpho-syntactic) categories, and elicitation methods. These minor objects can be used to assign values to the attributes of the major objects discussed above; e.g., tags can be used to classify forms, files, and collections while categories are relevant to forms only. Administrators (i.e., contributors with additional privileges) can specify possible grammaticality values as well as inventories of graphemes, phones, phonemes, and morpheme delimiters that can be used to configure input validation and orthography conversion. Input validation means restricting (if desired) which characters (and sequences of characters) can be used in specific form transcription values, i.e., orthographic and (narrow and broad) phonetic transcriptions and morphophonemic segmentations. This contributes further to consistency of representation and thus can be seen as a benefit to search. Orthography converter

2.1. Overview of the OLD

specification entails defining simple mappings between grapheme inventories so that contributors may interact with (i.e., create, modify, browse, search) the data using their desired orthography while maintaining consistency of underlying orthographic representation.

Accessing an OLD application requires authentication via a valid combination of username and password. Authorization to make particular requests is determined by user role, one of viewer, contributor or administrator. Viewers have read-only access to an OLD application's data, contributors can create, modify and delete objects, and administrators can do anything the system allows. Using this authentication/authorization system, producers of an OLD application can prevent public access to data and can permit certain users read-only access without opening up the possibility of accidental or malicious data corruption.²⁴ A further data privacy feature is implemented via a special *restricted* tag on forms and other objects; objects so tagged cannot be accessed by users unless they are administrators or are users who are themselves tagged as *unrestricted*. Privacy and access to data are discussed further in section 2.3.11. A more sophisticated access restriction system may be required if an OLD is to act as envisioned here, i.e., as a web service providing data to other specialty applications.

The OLD provides powerful search functionality, the more modest goal therefrom being timely access to relevant data and the more ambitious the discovery of novel generalizations. A search query is a tree structure of filter expressions whose nodes are conjunctions or disjunctions. There is no

²⁴OLD form and collection data are also versioned, i.e., backed up upon each modification. This is a further safeguard against data corruption.

principled limit on the complexity of this structure. A filter expression is an assertion that the value of a specified attribute matches (in a specified manner) a specified search pattern. Methods of matching between pattern and value include exact match, substring match and regular expression match. Filter expressions may also be negated. The end result is that forms (and other core objects) can be filtered with a high level of precision and relevant data can be quickly accessed. These search queries are themselves OLD objects that can be saved for later reuse or as the basis for modified queries or for defining corpora.

Of notable promise is the potential to create complex queries that use regular expression patterns to match system-supplemented morphological representations and structural patterns to match user-created syntactic representations. Section 2.3.6 discusses the search functionality of the OLD in detail and provides examples of practical queries.

Some remarks on the relation between the OLD software and language-specific OLD applications may be useful here. A language-specific OLD application is simply a web application that was created by installing the OLD software on a server and configuring it for use on a particular language. Configuring the OLD for use on a particular language does not require modifying the source code since the OLD was designed to be used on any language. Language-specific configuration involves setting up a domain name, identifying the language being documented (along with its ISO 639-3 identifier, if desired), and possibly also specifying language-specific orthographies/inventories, configuring validation and/or orthography conversion, specifying tags, syntactic/morphological categories, morpheme de-

2.1. Overview of the OLD

limiters, possible grammaticality values, etc., and creating HTML pages to guide users in following the conventions of the application. This type of configuration allows the OLD to be useful within a wide range of linguistic conventions and language patterns.²⁵ That said, field linguists working on, for example, sign languages, languages with right-to-left writing systems, or languages where the space character is not used to separate words may have requirements for a linguistic fieldwork application that the OLD does not, at present, meet. As I cannot anticipate all possible use cases, I hope that fieldworkers will continue to use the OLD and challenge its assumptions so that it can evolve to be a useful tool to an even wider range of languages and linguistic conventions.

This section has provided an overview of the OLD, showing how the system is used to create fieldwork-relevant objects that can be displayed, exported, and searched in ways that are beneficial to fieldworkers.

2.1.4 Implementation

Both versions of the OLD are open source and are written in the Python programming language using the Pylons web framework.

Version 0.2 is a standard Pylons web application whose design adheres closely to the recommendations and example application of *The Definitive Guide to Pylons* (Gardner, 2008). It is built in accordance with the model-view-controller (MVC) design pattern (cf. Krasner et al., 1988). The model

²⁵One hard-coded assumption of the OLD is that the space character will be used to delimit words. Based on this assumption, and the morpheme delimiters specified by administrators, the system creates column-aligned interlinear glossed text (IGT) representations and identifies words and morphemes for internal cross-referencing.

2.1. Overview of the OLD

is an SQLAlchemy-based Python interface to a MySQL relational database. The views are HTML pages generated server-side using the Mako template library. The controllers are Pylons-based Python objects that mediate between user actions on the view interfaces and database queries facilitated by the model. While some of the interface logic is JavaScript-coded and executed client-side and while some requests to the server are made asynchronously, the bulk of the user interface is implemented via requests for server-generated HTML pages. This results in the ungainly page refreshes and generally sub-optimal user experience typical of older generation web applications. Moreover, since OLD 0.2 server responses are HTML pages and client requests are standard Hypertext Transfer Protocol (HTTP) form submissions, disentangling the core application logic from the user interface in order to re-purpose the former for, say, an audio dictionary, language-learning software, or command-line interaction, would be prohibitively difficult.

While version 1.0 of the OLD is still written using the Pylons web framework, the user interface has been removed and replaced with a standards-compliant (RESTful, Fielding (2000)) Application Programming Interface (API). All incoming and outgoing communication is formatted as JavaScript Object Notation (JSON)²⁶ and the semantics of a request is determined by the HTTP method used. The effect of this is that multiple interfaces may be used to interact with a single OLD 1.0 web service. Currently under development is a single-page browser-based JavaScript application that replicates the functionality of the OLD 0.2 graphical interface while providing a richer

²⁶<http://www.json.org>

user experience. However, I envision additional interfaces to the OLD 1.0 web service. Already constructed is a simple Python module that uses the Requests library to interface with a live OLD application via HTTP;²⁷ this has been put to good use in the creation and evaluation of morphological parsers (as detailed below). Other possibilities include mobile platform or traditional desktop interfaces, as well as interfaces with other web services/applications such as talking dictionaries or language learning games.

The OLD 1.0 is licensed under the Apache License, Version 2.0 and its source code can be found on GitHub. The OLD 0.2 is licensed under the GNU General Public License, Version 3 and its source code can also be found on GitHub.

2.2 Other fieldwork tools

This section reviews a subset of existing fieldwork software tools. Those reviewed were selected because they are widely used and/or are similar to the OLD. The intention here is not to demonstrate that the OLD is superior to these tools in all respects but to explore the strengths, weaknesses, and biases of each tool, to discover innovative and useful features that the OLD may borrow, to explore how these tools can complement one another and be used in conjunction, and, of course, to argue for the benefits of the OLD in certain respects. Reviewed here are the two products of SIL International that are probably most widely used by linguistic fieldworkers, i.e., The Field Linguist's Toolbox (§ 2.2.2) and FieldWorks Language Explorer

²⁷See the `oldclient.py` module at <https://github.com/jrwdunham/old-parser-research>.

2.2. Other fieldwork tools

(FLEx, § 2.2.3). LingSync, a recently released web application for linguistic fieldwork, is discussed in § 2.2.4. Prior to these discussions, § 2.2.1 explores the role of SIL International in fieldwork and endangered language documentation and the relation of the OLD to that organization and the software it has put out.

There are, of course, many other software tools not discussed here that are designed for, and used in, linguistic fieldwork. The online journal *Language Documentation & Conservation (LD&C)* has published reviews on many of them. Web-based, general-purpose fieldwork tools include Em-dros (Lowery, 2008), TypeCraft (Farrar, 2010; Beermann and Mihaylov, 2010), and LEXUS (Kotcheva, 2009; Ringersma and Kemps-Snijders, 2010). Tools that focus on annotation of digital audio and/or video files include ELAN (Berez, 2007; Hellwig et al., 2013), CLAN (Meakins, 2007), Audiamus (Brotchie, 2007), InqScribe (Garde, 2012), Anvil (Tan and Martin, 2011), Transcribe! (Barwick, 2009), Transana (Afitska, 2009), and EXMARaLDA (Meißner and Slavcheva, 2013). Geared primarily toward the production of dictionaries are Kirrkirr (McElvenny, 2008), TshwaneLex (Bowern, 2007), and WeSay (Perlin, 2012). Other relevant tools include The Linguist's Assistant (Beale, 2012) and Phon (Buchan, 2011). Finally, many fieldworkers use spreadsheet applications, general-purpose desktop database applications (e.g., Microsoft Access or FileMaker Pro), or (increasingly) their web-based counterparts (e.g., Google Spreadsheets) which offer similar functionality while facilitating multi-user web-based collaboration.

Table 2.2 compares the salient features of the fieldwork applications discussed in the sections that follow. In general, Toolbox and FLEx place em-

2.2. Other fieldwork tools

	OLD	Toolbox	FLEx	LingSync
dictionary ^a	partial	yes	yes	no
texts ^b	yes	yes	yes	no
IGT ^c	yes	yes	yes	yes
search	yes	partial	yes	yes
parser ^d	yes	yes	yes	partial
web collaboration ^e	yes	no	partial	yes
web API	yes	no	no	yes
platforms ^f	WML	W	WL	WMLA
open source	yes	no	yes	yes

^a Support for creating dictionaries.

^b Support for creating texts.

^c Interlinear glossed text (IGT) data display.

^d Morphological parser creation function.

^e Web-based, multi-user collaboration.

^f Platform support: W=Windows, M=Macintosh, L=Linux, A=Android.

Table 2.2: Comparison of fieldwork software.

phasis on the creation of dictionaries with supporting texts; FLEx can be viewed as an upgraded Toolbox with improvements in terms of search, multi-user collaboration, cross-platform support, and an open source software license. The OLD and LingSync are open source web applications that focus on multi-user collaborative database creation and cross-platform operability. See the reviews of these tools below for justification of the values in the cells of Table 2.2.

2.2.1 SIL International

SIL International (SIL) is a global organization that for the past eight decades²⁸ has made undeniably substantial contributions to the documentation and description of endangered languages. However, because one of its three primary goals is Bible translation²⁹ and because of its close association with its partner organization Wycliffe Bible Translators, whose mission is “to see a Bible translation program in progress in every language still needing one by 2025” (Wycliffe, 2013), many academic linguists are uncomfortable with being *de facto* dependent (Dobrin, 2009) on the organization for a wide range of documentation-related tasks.

It is undeniable that SIL is a world leader in endangered language documentation. To give a sense of the size and output of the organization, consider that SIL International has linguists in “more than ninety countries [...]”, includes approximately 6,500 members, [...] recently celebrated the completion of its five hundredth translation of the New Testament, [has ongoing work in] over 1,100 other languages” (Svelmoe, 2009), has a bibliography of “over 13,000 [...] books, journal articles, book chapters, dissertations, and other academic papers” (Olson, 2009), and “reported over \$41,000,000 in revenue on its 2005 tax return” (Epps and Ladley, 2009). Not only do SIL linguists produce grammars, dictionaries, and collections of texts that are crucial to academic linguistics, they also produce (or have leading roles in the production of) core documentation-relevant technologies and

²⁸The first Summer Institute of Linguistics (SIL) was held in 1934 in Arkansas. For a concise history of the organization see Svelmoe (2009).

²⁹According to Olson (2009), the other two are (1) research and training of linguists, and (2) literacy and education.

2.2. Other fieldwork tools

standards. These include fieldwork-facilitating software like The Linguist’s Shoebox and FieldWorks Language Explorer, the keyboard layout editor Ukelele, fonts (e.g., Doulos, Charis, and Gentium) for representing the rare characters used in endangered language transcriptions (SIL International, 2013a), a comprehensive catalogue of the world’s languages (i.e., Ethnologue), and (derived from the previous) the internationally recognized ISO 639-3³⁰ standard code set for identifying the world’s natural languages.³¹

Many academic linguists are uncomfortable with the level of influence that SIL has in the field of language documentation. Works such as Dobrin and Good (2009) can be considered a call to action for the academy to take on greater responsibility in this area. Some go further and argue that the mere presence of wealthy SIL Bible translators in impoverished indigenous communities exerts a powerful proselytizing effect and inevitably works to corrode culture and language (Epps and Ladley, 2009), even despite protestations (cf. Olson, 2009) that SIL is not a missionary organization and that its members are explicitly prohibited from preaching, baptizing, and creating churches. Olson (2009) views the academic distrust of SIL as thinly veiled condescension toward indigenous peoples’ ability to determine their own cultural, religious, and linguistic practices.

Handman (2009) argues that secular linguists must acknowledge a de-

³⁰ISO, the International Organization for Standardization, is “an international standard-setting body composed of representatives from various national standards organizations” (Wikipedia, 2013).

³¹There is debate in the linguistic community over whether and how linguists should use the ISO 639-3 standard. Morey et al. (2013) represents a highly critical view. Good and Cysouw (2013) argues that the ISO 639-3 standard is useful in certain cases and proposes a complementary inventory of “doculects”, entities that are rigorously defined according to sets of sources, i.e., documentary artifacts, and upon which such concepts as *language* and *dialect* may be defined. See Glottolog at <http://glottolog.org/>.

structive ideology of their own, one which seeks to preserve a perceivedly authentic cultural homogeneity that is at odds with the reality of cultural change and exchange. A response to this charge can be seen in the exhortation of Amery (2009) that linguists interested in the future value of their work for revitalization elicit more natural conversations containing common usage and move away from attempting to gather only “pure”, traditional forms, unaffected by contact.

The strongest argument against SIL vis-à-vis endangered language documentation is, to my mind, the claim that the organization tends to strategically move resources and focus away from the most highly endangered languages—“precisely [those] that academic linguistics now deems most urgently in need of attention” (Dobrin and Good, 2009, p. 624)—since they expect that soon there will be no readership for Bibles produced in those languages (see also Epps and Ladley, 2009). While a valid concern, this alone clearly does not preclude collaboration between SIL and the wider community of linguists.

Clearly academic linguists and members of Protestant Bible translation organizations like SIL have different high-level goals yet share many common interests. Even those who have no issue with collaborating with SIL and using the technologies generated by that organization should recognize the value in the independent development of tools like the OLD. The features provisioned by such tools have the potential to have a direct and positive impact on the progress of linguistic research, language documentation, and community-based revitalization. The engineering of such tools requires a creative synthesis of developer, researcher, documentor, and educator ex-

expertise. As technologies evolve and linguistic landscapes change, we need to be able to adapt in accordance with our own goals.

On this topic, it is worth quoting Dobrin and Good (2009) at length:

For linguistics to externalize the development of technological and community resources because the problems they solve are practical rather than scholarly, or because they are others' rather than ours, is increasingly untenable. The problems that call upon specialized linguistic knowledge for their solution are numerous and growing, and indifferent to the traditional boundaries of the discipline. [...] Linguistics could come to more closely resemble fields like medicine or economics, where interplay between theory and practice is welcomed as adding to their richness, and where 'applied' forms of work are not seen as belonging to a separate discipline (Dobrin and Good, 2009, pp. 628–629).

The OLD is, in my estimation, a viable alternative to SIL's fieldwork software tools The Linguist's Shoebox, The Field Linguist's Toolbox, and FieldWorks Language Explorer. However, issues with the organization itself (ideological or otherwise) were not motivating factors in its construction. Rather these motivations were primarily technical and methodological in nature: the fieldwork software offerings from SIL are focused on a single platform (viz. Windows),³² are not open to programmatic alteration by outside developers,³³ and do not support web-based multi-user contribution as

³²Shoebox has a PowerPC Mac implementation, but that architecture is all but obsolete. Toolbox has no native Mac or Linux implementations. FLEx can be used on Linux platforms but not Macintosh ones.

³³While FLEx is open source, Shoebox and Toolbox are not.

a core feature.³⁴

2.2.2 Toolbox

Perhaps the best known and most widely used linguistic fieldwork software application is The Field Linguist's Toolbox (and its predecessor The Linguist's Shoebox), developed by SIL International.³⁵ This freely available desktop application is designed to help linguists and anthropologists organize and process their linguistic fieldwork data. Though it has advantages over the OLD, some disadvantages are that that it is closed source, focused on a single platform (Windows), and not primarily concerned with the facilitation of web-based collaboration. This section describes Toolbox with reference to its own documentation and a published review and addresses how it compares to the OLD.

The following description of Toolbox is quoted from SIL International (2013d).

Toolbox is a data management and analysis tool for field linguists. It is especially useful for maintaining lexical data, and for parsing and interlinearizing text, but it can be used to manage virtually any kind of data. ... Toolbox also has powerful linguistic functionality. It includes a morphological parser that

³⁴Shoebox and Toolbox do not provide this functionality. FLEx does introduce it, though Butler and van Volkinburg (2007) reports crashes when this feature is employed. I have yet to experiment with FLEx's support for multi-user concurrent access.

³⁵Technically, Shoebox is on SIL's list of supported products whereas Toolbox is endorsed but "not necessarily support[ed]" (cf. <http://www-01.sil.org/computing/catalog/>). According to Dobrin and Good (2009, p. 623, fn.6), "Individuals associated with SIL have continued to develop Shoebox under the new name 'Toolbox', but these activities do not represent official efforts on the part of SIL's computing division."

can handle almost all types of morphophonemic processes. . . . It has a user-definable interlinear text generation system which uses the morphological parser and lexicon to generate annotated text. Interlinear text can be exported in a form suitable for use in linguistic papers. Toolbox has export capabilities that can be used to produce a publishable dictionary from a dictionary database.

The OLD can also be used to maintain lexical data. Using the corpus and search objects, any number of distinct lexica may be defined. OLD applications also facilitate the generation of interlinear texts—in part by allowing for the creation of an unlimited number of morphological parsers—and these interlinear texts can also be exported in forms suitable for use in papers. Version 0.2 of the OLD has a limited dictionary interface to forms deemed lexical by the system according to a simple heuristic. Improvements to the dictionary-creation functionality of the OLD—including building on lexical corpora to create dictionary objects exportable to XeLaTeX/PDF—are discussed in § 2.3.9.

The review of Toolbox in Robinson et al. (2007) extols its flexible data structure and its easily comprehensible and programmatically manipulable storage format. The major criticisms from that source are the difficulty in the initial setup of a Toolbox project, issues with auto-synchronizing lexica and interlinear glossed text documents, weaknesses in supporting data consistency, and a lack of an integrated scripting language for automating functions.

OLD applications are not, in my estimation, difficult to set up. However,

2.2. Other fieldwork tools

this is probably due to the more rigid nature of the OLD data structure in contrast to the more flexible Toolbox one. Installing and serving an OLD application does require some basic proficiency with the command-line and server configuration; however, this is facilitated by mature package management tools (i.e., EasyInstall and pip) and is thoroughly explained in the documentation (Dunham, 2013a). Once an OLD web service is being served online and has a valid domain name, the actual setup is quite minimal. It involves specifying the object language name, creating some user accounts, and specifying some optional configuration settings, e.g., inventories for input validation.

Because the OLD prescribes a particular structure for fieldwork data (cf. § 2.3.4), initial configuration does not require data structure specification, i.e., a potentially complicated setup of a hierarchy of fields with their own data types. The disadvantage of the OLD approach is, clearly, that the data structure can only be adapted to a limited extent to the desires or existing structures of the fieldworkers planning to use it.³⁶ Existing data must be converted to an OLD-compatible format and imported before it can be used within an OLD application.³⁷ Interested readers should consult § 2.3.4 in

³⁶In my experience, idiosyncratic aspects of fieldworkers' data structures can usually be accommodated within the OLD structure via the general purpose tagging mechanism. There is, for example, no need to have dedicated database columns (or the equivalent) for simple attributes like *non-future tense semantics* or *need to re-licit* since this information can be encoded via tags. Even multi-valued attributes such as *semantic field* can be encoded as OLD tags by establishing conventions within the tag names, e.g., *sem fld: mankind* or *sem fld: animals*. Forms can then be searched using regular expressions to retrieve results with a specific set of semantic field values. Other types of information can be stored in the general comments attribute of forms and, if necessary, syntactic conventions can be established to facilitate retrieval.

³⁷Future versions of the OLD may allow users to extend the data structure via the interface.

order to determine whether the OLD data structure is sensible, general, and flexible enough to handle their fieldwork needs. Of course, since the OLD is open source, it is entirely possible for interested parties to modify the data structure directly (though this will require some working knowledge of the technologies used).

The OLD version 0.2 suffered from lexicon-text synchronization issues similar to those attributed to Toolbox in Robinson et al. (2007). In that version, HTML representations of collection objects (used to generate texts containing IGT-formatted language data) are inefficiently generated anew upon each read request. In the OLD 1.0 revision, HTML representations of collections are now generated upon each successful create/update request and are never generated during a read request. This is more efficient since texts (i.e., collections) are more commonly read than modified. However, now whenever a form is updated, the HTML representation of all collections containing it are automatically updated as well. This means that texts (i.e., collections) are always synchronized with the forms that they reference. In addition, the form-referencing attributes of forms (i.e., syntactic category string, break-gloss-category, morpheme break identifiers, and morpheme gloss identifiers; see Appendix A) are also updated when an implicated form is modified (see the next paragraph and § 2.3.10 for details).

In response to the criticism of Toolbox by Robinson et al. (2007) with respect to supporting data consistency, note that the OLD does this in a number of ways, as detailed in § 2.3.8. One such consistency-enforcing feature is effected via the use of relational objects in the attributes of forms.

That is, if, say, the name value of an elicitation method object is changed,³⁸ then all forms associated to that elicitation method reflect the change immediately and consistently. This is one of the advantages of the relational data structure: the attributes of numerous objects may receive their values from a single object, thus allowing modifications on one to percolate to all the others associated to it (see § A.1). Where necessary, non-relational values (e.g., strings) are modified upon update of a relevant related object. For example, if a contributor changes the name of a *syntactic category* object, the system will automatically update the *syntactic category string* value of all forms whose morphological analyses contain a morpheme of that category.³⁹

Like Toolbox, the OLD does not include an integrated scripting language for automating tasks (cf. Robinson et al., 2007). Such a feature might be useful for when users wish to make widespread modifications to the data set based on a series of conditions and context-dependent transformations, e.g., finding all forms glossed with a certain morpheme and changing the gloss to one of a number of possibilities, as determined by its neighbouring glosses. Other use cases might be programmatically retrieving sets of minimal pairs or statistics on morpheme frequency. However, since the OLD is a web service that communicates via JSON-encoded data structures across HTTP, such tasks can be accomplished using *whatever programming language the so-inclined contributor wishes to use*. For example, one could write a Python (or Perl, C++, Java, etc.) script that issues the appropriate HTTP request

³⁸In the OLD, elicitation methods are objects (i.e., entities with named attributes) in their own right. They have two primary attributes: *name* and *description*.

³⁹See § 2.3.6 for a detailed exposition of the use and method of generation of syntactic category string values and related attributes.

2.2. Other fieldwork tools

to retrieve all forms in an OLD application matching OLD search criteria, then performs the modifications client-side, and then issues a series of HTTP update requests to alter the server-side data. The possibilities are endless. That said, the particular feature of being able to perform complex system-wide updates via a dedicated interface (i.e., “bulk update”) has been frequently requested and will probably be implemented in future versions of the software.

The use of JSON as the OLD’s communication format touches on the commendation in Robinson et al. (2007) of Toolbox’s data storage format as simple to understand and manipulate. All OLD 1.0 communication is JSON. JavaScript Object Notation is a widely used standard in the programming world for creating string representations of commonly used data types⁴⁰ and most major programming languages have at least one library for converting between native data structures and JSON. This makes it very easy for interested fieldworkers to access and programmatically manipulate OLD data.⁴¹

Relative to the OLD, Toolbox has better support for dictionary creation and boasts a more flexible data structure that users can tailor to their needs. On the other hand, the OLD has arguably better search capability, has better facilitation of data consistency, allows for the integration of media files, is open source, is web-based, and, as a result, has better cross-platform coverage. The differences between these two tools can be traced back to a

⁴⁰JSON can encode JavaScript objects (i.e., associative arrays or dictionaries), arrays (i.e., lists), strings, numbers, and Booleans.

⁴¹Those with back-end access to an OLD application can also interact with the data more directly via Python or Structured Query Language (SQL) (through MySQL or SQLite).

fundamental difference in the type of fieldwork and fieldworker targeted. Toolbox helps fieldworkers who are primarily interested in documentation and description to create lists of lexical items, dictionaries, and interlinear glossed texts. The OLD is arguably more general in that it can be used to facilitate both language description and theoretical linguistic research. This same contrast becomes apparent when comparing FLE_x with the OLD and is discussed further in § 2.2.3 below.

2.2.3 FLE_x

This section discusses FieldWorks Language Explorer (FLE_x) and compares it to the OLD. Since this application is being actively developed and promoted by SIL International, it is reasonable to assume that it has an extensive user base and that many readers will be familiar with the FLE_x (and Toolbox/Shoebox) approach to computational language documentation and description. I therefore devote considerable attention to this tool in order to illuminate the ways in which it differs from the OLD and the reasons for those differences. There are several ways in which the OLD could be improved by emulating the functionality of FLE_x. On the other hand, the OLD is, as argued below, superior to FLE_x in a number of key areas.

FLE_x is SIL International's currently supported, general-purpose fieldwork tool. It is a major component of the suite of tools called FieldWorks.

FieldWorks Language Explorer (or FLE_x, for short) is designed to help field linguists perform many common language documentation and analysis tasks. It can help you elicit and record lexi-

2.2. *Other fieldwork tools*

cal information, create dictionaries, interlinearize texts, analyze discourse features, [and] study morphology (SIL International, 2014b).

It boasts a sophisticated graphical user interface and data structure specifically tailored for the creation and maintenance of lexical entries to be assembled into dictionaries. The system also has impressive support for creating interlinear texts, including features for ensuring that these are consistent with the lexicon, for configuring morphological parsers to automate the generation of morphemic analyses, and for identifying phrasal constituents and labelling them according to system-provided (and user-modifiable) grammatical roles, e.g., subject, verb, clause in object position, etc. (SIL International, 2014a). In addition, FLE_x provides powerful search/filtering and bulk editing functionality, is freely available and open source (MIT License, written in C#), and runs natively on Windows and Linux platforms (though no native Macintosh version is currently available.)

SIL's development of the FieldWorks suite represents a transition from the lexicon creation and text analysis focus of Shoebox toward "translation-related tasks for which there have been fewer computer solutions" (SIL International, 2013c). However, from what I can gather, FLE_x itself is still focused primarily on the creation of a lexicon and supporting interlinearly analyzed texts. Consider in this regard the description of the software in Black and Simons (2008, p. 37) as "the lexicon and text component of the SIL FieldWorks suite of tools." The primary tools for assisting with translation-related tasks are ParaText and (the now deprecated) Translation Editor.

2.2. Other fieldwork tools

I will begin by discussing FLEx’s cross-platform support. I successfully installed the FieldWorks suite version 7.0.6 on Ubuntu 12.04.3 (Precise Pangolin) following the instructions at its Linux developer site,⁴² which is linked to from the main FieldWorks download page. SIL International’s Language Software Development division’s Linux team (LSDevLinux) is working to port FLEx to Linux and Mac operating systems. At the time of writing, the latest stable release of FieldWorks for Windows is 7.2.7 while version 7.0.6 is the one that is available from LSDevLinux for Linux. Note that my computer runs Mac OS X 10.6.8 (Snow Leopard) and I was able to install the FieldWorks suite on the (open source) Ubuntu Linux operating system that was itself installed on my Mac via the (open source) virtualization software package VirtualBox.⁴³ The installation went relatively smoothly and was accomplished entirely through graphical interface tools, i.e., no command line. Therefore, through some not insignificant level of indirection (i.e., Mac to VirtualBox to Ubuntu to FLEx), I was able to run FLEx on a Mac using open source and freely available technologies throughout. I cannot provide a comparison of the relative performance of FLEx on top of such a technology stack, but I suspect that it would be degraded using this, or a similar, virtualization approach. That is, a native Mac version of the software is still something to be desired.

The most salient difference between FLEx and the OLD is that the former is geared toward practical lexicography. FLEx’s default data structure

⁴²linux.lsdev.sil.org/wiki/index.php/FieldWorks_Installation_Instructions

⁴³As explained on <http://en.wikipedia.org/wiki/VirtualBox>, VirtualBox is a “virtualization software package [that] is installed on an existing host operating system as an application; this host application allows additional guest operating systems, each known as a Guest OS, to be loaded and run, each with its own virtual environment.”

(though customizable) and its interface conveniences are centred around the creation of lexical *entry* objects for dictionaries. The primary field (i.e., attribute) of an entry is the *lexeme form* and entries may have multiple *senses*, each with their own *gloss*, *definition*, *category*, and *example sentence* values (cf. SIL International, 2014a). The system recognizes and appropriately displays (using dictionary formatting conventions) homographic entries as well as distinct senses of a given entry. For each sense of an entry, users can specify the category (e.g., N) as well as inflectional features (e.g., neuter gender). Affixal entries can be specified for morpheme type (suffix, prefix, infix, proclitic, etc.), affix type (inflectional or derivational), the category they attach to, and the categorial change (if any) that they effect. For each entry, multiple allomorphs and their contexts of occurrence can be specified. This information is used to configure the morphological parser. Users can specify a variety of relations between entries which can affect how they are displayed within the dictionary view; these relations include synonym/antonym of, part-to-whole, and variant of. Morphologically complex entries can be described by decomposing them into existing entries and the system can be configured to display these complex entries under the simplex headwords in various ways. Figure 2.1 repeats the examples used by SIL International (2014a) to illustrate dictionary representations generated from FLEx entry objects. Shown here are homographs (*bank*₁ vs. *bank*₂), morphologically complex entries (*blinder*), variants (*blinker*), and semantically (*wood*) and categorially (*free*) ambiguous lexical entries.

The FLEx interface also provides a number of conveniences for constructing lexical entries. These include built-in detailed and hierarchically

bank₁ *N* financial institution
bank₂ *N* edge of river
blinder (der. of **bind**, **-er**, dial. var. of **blinker**)
N either of two flaps on a horse's bridle to keep it from seeing objects at its sides
free 1) *Adj* unrestricted 2) *V* to relieve from restrictions
wood *N* 1) A dense growth of trees usually greater in extent than a grove and smaller than a forest. 2) The hard fibrous substance used to make furniture etc.

Figure 2.1: Dictionary-style representations targeted by FLEx (cf. SIL International, 2014a).

organized analytic information to help with choosing things like grammatical categories and inflectional features for senses. The *categorized entry* interface guides language documentors in creating a broad-coverage dictionary by providing a hierarchy of semantic domains as suggestions for entries to elicit and then providing a simplified input form for quickly creating lexical entries that can later be refined via the standard entry interface.

In contrast to FLEx, the OLD assumes a user base that is engaged primarily in the creation not of dictionaries and texts but of sets of forms that are relevant as evidence for or against particular theoretical linguistic claims. This is particularly apparent when one considers that theoretical linguists are often interested in discovering the grammaticality of specific constructions and, as a result, end up collecting ungrammatical forms which really have no place in lexica, dictionaries, or texts as representations of utterance events. Unlike a FLEx *entry* which may have multiple allomorphs

and multiple senses, each sense with its own gloss and grammatical category, an OLD *form* has a single shape, a single gloss, and a single category. Therefore, additional efforts would be required in order to achieve the lexicographic structure of a FLEx database within an OLD one. One approach would be to designate certain OLD forms as entries and then create references to the other forms whose phonemic, semantic, and categorial information are to be used in the specification of allomorphs and distinct senses for said entries. Using the current OLD data structure (cf. § 2.3.4), this could be accomplished by creating a *lexical entry* tag and then establishing syntactic conventions for referencing other form objects within, say, the general comments value of the form-as-entry. For example, a convention could be established according to which inserting the strings *allomorph*[37], *allosense*[899], and *antonym*[1123] into the general comments value could constitute a reference to other forms whose data could be used to assemble information on allomorphs, distinct senses, and antonyms, respectively of the lexical entry form.⁴⁴ Other aspects of the FLEx data structure could be encoded within the current OLD data structure via the creation of tags for inflectional features, morpheme types, affix types, derivational morpheme categorial inputs and outputs, etc. In order to display this information via an appropriate dictionary-type representation, an interface to an OLD web service would, of course, need to be aware of any such conventions.

However, it may be better strategy to focus on how the OLD can be made

⁴⁴String-based references to other forms within the general comments value is a bit of a hack. A more robust solution would be to alter the data structure to allow for different types of relations between forms and other forms, e.g., endowing form objects with sense attributes that can refer to zero or more other form objects.

interoperable with dictionary-creation software and/or standards—e.g., by implementing appropriate import and export features—rather than focusing on how the OLD can be modified to *replicate* the data structures and feature sets of such software. Instead of engaging in a futile wheel-reinventing arms race, I anticipate that future work on the OLD will stick to refining its strengths—viz. data-sharing, collaboration, online accessibility, catering to theoretical linguistic analysis, computational implementation of linguistic models—while adding the import/export functionality that is crucial for complementary co-existence with extant useful tools such as FLEx.

It is interesting to note that the differences between the OLD and FLEx data structures reflect differences in fundamental analytic assumptions about the lexicon and other components of the grammar. The OLD assumes that the lexicon is a set of morphemes that are unique form-meaning-category triples. Allomorphy is to be encoded in the phonological component as transformations on phonemic shapes of morphemes. For example, the surface realization of the English morpheme /in/ ‘not’ as [im] before labials is something that should be encoded within a phonology, i.e., an OLD phonology object (cf. § 3.3).⁴⁵ Similarly, the categorial and semantic ambiguity of FLEx lexical entries would, under the assumptions underlying the OLD, be better analyzed as distinct morphemes. For example, the English noun *dog* ‘canine’ and the verb *dog* ‘follow’ are distinct morphemes. If a researcher wishes to analyze the verb as derived from the noun via a phonologically null derivational affix, that is certainly representable within an OLD application.

⁴⁵OLD phonologies can even capture suppletive transformations since the phonological rules can be made to be sensitive to phonemic, semantic, and categorial information. For example, an OLD phonology could map /be-z/ ‘be-3.sg’ ‘V-Agr’ to [was], cf. § 3.3.

2.2. Other fieldwork tools

Turning to FLEx’s bulk edit functionality, we find an impressive array of features that could be emulated by software like the OLD to good effect. When bulk editing, users first create a filter to select the subset of entries to which the edit should apply, then configure the transformation on their chosen field, optionally indicate *ad hoc* exceptions to the transformation, preview the effect of the bulk edit, and then apply, if desired. The transformation may insert a particular value into the column⁴⁶ of all filtered rows, copy a row’s value for one column into another, replace all column values with a specified value, delete the content of a selected column, delete an entire entry, or specify a predefined process to change the value of a column or copy a changed value to another column. The process illustrated in SIL International (2014a) maps a Devanagari transcription to an IPA one. Users can also specify what the system should do when the target column of a bulk edit already contains a value: do nothing, overwrite it, or append to it.

Using FLEx, users can create any number of interlinearly analyzed texts. Texts are created by first inserting a transcription into the so-called *baseline* view. The system then provides the *analyze* interface for specifying the morphological analysis.⁴⁷ Via this interface, users can specify a segmentation into allophones⁴⁸ (the *morphemes* row), a segmentation into lexeme citation forms, lexeme glosses, lexeme grammatical categories, word glosses,

⁴⁶In this discussion I am using the terminology of *columns* and *rows* in accordance with the tabular representation of entries provided by FLEx in the bulk edit interface. That is, each row represents an entry and each labelled column represents an attribute (or field) of all entries.

⁴⁷Technically there is another *gloss* interface for specifying morphological analyses. The fields of the more complex *analyze* interface are a superset of those of the gloss interface.

⁴⁸Note that this is different from the OLD where there is no dedicated attribute for allophones. This reflects a fundamental difference in how morphological analyses are represented and in how parsers are configured by the two systems. See the discussion below.

2.2. Other fieldwork tools

word categories, and a free translation.

An illustration of how a particular analysis of the French sentence *Les chiens courraient* might be represented within a FLEEx text is provided in (1). Note that the system requires that the segmentation into morphemes in the *morphemes* field be constructible by inserting one or more morpheme delimiters into the *word* value; that is, one could not segment *courraient* into, say, *courir-aient*, i.e., the verb meaning ‘run’ in its infinitival form followed by a tense-aspect/person agreement suffix. Also note that the values for *lexeme entry*, *lexeme gloss*, and *lexeme grammatical info*⁴⁹ for a particular morpheme *must* correspond to values of a lexical entry that is already listed in the lexicon; if the lexicon contains no matching entry, the interface allows for the specification of one without leaving the *analyze* interface. Note also that if the *morphemes* value of a given column matches an *allomorph form* of an existing lexical entry, FLEEx will recognize this and populate the *lex. entries*, *gloss*, and *gramm. info.* fields with that entry’s values, as appropriate.

(1)	WORDS:	<i>Les</i>	<i>chiens</i>	<i>courraient.</i>		
	MORPHEMES:	le -s	chien -s	courr	-aient	
	LEX. ENTRIES:	le -s	chien -s	courir	-aient	
	LEX. GLOSS:	the pl	dog pl	run	3.pl.ipfv	
	LEX. GRAMM. INFO.:	d x:{d,n}	n x:{d,n}	v	v:v	
	WORD GLOSS:	the	dogs	were	running	
	WORD CAT.:	d	n	v		
	FREE TRANSLATION:	‘The dogs were running.’				

⁴⁹The lexeme grammatical info field of a FLEEx entry can specify the category of the lexeme or the categories of the morphemes to which it can affix and the category of the resulting morphologically complex unit. Thus *x:{d,n}* in (1) indicates that the plural morpheme can suffix to nouns or determiners (*{d,n}*) and that the category of the resulting complex is that of the free morpheme (*x*). The way that I have represented this information, however, may not accurately reflect the syntax accepted by FLEEx.

2.2. Other fieldwork tools

Contrast (1) with an analogous analysis of the same sentence as represented by the OLD in (2).

(2) WORDS: *Les chiens courraient.*
MORPHEME BREAK: le-s chien-s courr-aient
MORPHEME GLOSSES: the-pl dog-pl run-3.pl.ipfv
CATEGORIES: d-num n-num v-agr
TRANSLATION: ‘The dogs were running.’

In the OLD representation in (2), there is no distinction between morphemes and lexemes. The system assumes that the morpheme shapes in the *morpheme break* value are phonemic transcriptions that can be mapped, via some phonological (or spelling) rules, to phonetic (or orthographic) transcriptions. Thus a researcher may segment *courraient* into *courr-aient*, *courrir-aient*, or anything else, as desired. If the user specifies a phonology that generates *courraient* from *courrir-aient*, then all the better; but this is not required. Neither is it required that the morpheme shapes and/or glosses used in the analysis correspond to lexical forms already present in the system. An OLD application provides visual feedback on the lexical consistency of a specified analysis (cf. § 2.3.5), but it will not enforce such consistency.

In the OLD, the *categories*⁵⁰ value is automatically generated by the system when the form is entered. If the system can find a form with *courr* as its *morpheme break/shape* value and *run* as its *morpheme gloss* value, then the category of that matching form (in this case *v*) will be inserted

⁵⁰Technically, this is the *syntactic category string* attribute of form objects. That attribute name is actually a misnomer since the categories are not necessarily syntactic. In future versions of the OLD, this attribute will probably be renamed simply to *categories*, the *syntactic category* relational attribute to *category*, and the *syntactic category* object to *category*.

2.2. Other fieldwork tools

into the auto-generated *categories* value of the larger form automatically. If no match is found, the category will be specified as *???*. Users cannot, at present, directly specify a value for the *categories* attribute of a form.⁵¹

Note also that, unlike FLE_x (cf. 1), the OLD provides no fields for specifying *word gloss* and *word category* values. Of course, this information might be implicit in the system, insofar as there exist form objects corresponding to the words of the multi-word form being entered. That is, if one enters (2), the system could, in principle, be modified so that it would check for word-level forms that match and use this information to generate values for these fields. In this example, the system would look for a form with a *morpheme break* value of *courr-ai-ent* and a *morpheme gloss* value of *run-3.pl.ipfv* and use this information to auto-generate *word category* and *word gloss* values using the *category name* and *translation* values of the matching form, e.g., *v* and *were running*. If this feature is implemented, these fields may also be made directly user-specifiable, as was discussed for the *categories* attribute above.

As discussed elsewhere in this dissertation with respect to morphemes, future updates to the OLD will include functionality such that whenever a user enters a multi-morphemic or multi-word form, the system will provide an interface that offers to create form objects for all of the morphemes *and words* that are implicit in the larger form being entered and which are not already present in the database. This will significantly increase the rate at which OLD data sets grow and will help researchers to generate lexica

⁵¹It may be desirable to allow users the option of explicitly specifying the *categories* value and have the system suggest a value using the method just described. This may be implemented in future versions of the OLD.

2.2. Other fieldwork tools

as a byproduct of eliciting sentential forms. This approach is also, in my judgment, superior to the FLEx approach which *requires* that lexical entries be present before they can be used in analyses of texts since it encourages consistency but allows the fieldworker to forego it for the sake of entering elicitation data quickly.

Before concluding this discussion of FLEx’s text interlinearization functionality, it should be noted that the word gloss and word category fields are not limited to words as defined by whitespace. That is, words can be grouped together and glosses and categories can be provided for these groups. This is illustrated in (SIL International, 2014a) by grouping *of course* into an idiomatic phrasal entry glossed as ‘obviously’ and categorized as an adverb. This is an interesting feature. At present, I am unsure of whether this is desirable or how it might be implemented within an OLD application.

Both the interface and the data structure of a FLEx project are modifiable. When viewing entries in tabular representation, users can configure the ordering of the columns used to display the entries and they can choose which columns are visible.⁵² Idiosyncratic fields can be added to entry objects as suits the user’s needs. Such a dynamic data structure is not available within OLD applications, although implementation thereof is a planned feature.⁵³ One particular type of data structure modification of FLEx that is

⁵²Note that I am here talking about the tabular view of multiple lexical entries and not the table-like IGT representation of sentences in a FLEx text.

⁵³The OLD data structure can be modified insofar as any number of named tag objects may be defined within an OLD application and any form, file, or collection may be associated to zero or more tags. Since tags may follow hierarchical naming conventions (e.g., *noun:num:pl*, *noun:num:sg*, *noun:gen:fem*, etc.) they allow for categorization and subcategorization of objects. Future modifications to make the data structure dynamic will allow for users to define string-valued attributes on core objects, such as forms, using the entity-attribute-value approach.

notable is the ability to subtype textual fields according to writing systems. Thus, for example, the gloss field of a sense of a lexical entry may be subdivided into English and Spanish subfields, thereby allowing a documentor to provide the same gloss for an entry in two distinct metalanguages.⁵⁴ Spell checkers can be integrated into the system and values that are misspelled according to their field's writing system can be discovered and corrected.

FLEx allows multiple contributors to collaborate on a single project across the Internet or within a local network. Contributors have a local copy of the data that they interact with via their FLEx desktop application and they can sync their local data with a master repository. Details of how this is implemented and how conflicts are handled when merging data sets are provided in SIL International (2013b). Butler and van Volkinburg (2007) attests to issues with the program crashing during multi-user concurrent access. However, that review is considerably dated and presumably many of the bugs with this feature have been worked out. Assuming that FLEx's collaborative functionality works as described, it must be admitted that the application allows for online/offline access to a communally created repository of fieldwork data.

Both FLEx and the OLD allow users to configure morphological parsers to facilitate the automatic morphemic analysis of complex words. However, the two applications differ in their conceptual modelling, computational im-

⁵⁴While the OLD allows a form to have any number of *translations*, each form can only have one *morpheme gloss* value. Therefore, a lexical entry in an OLD application cannot be simultaneously glossed in two distinct metalanguages. This is perhaps not a major failing of the OLD, but it does mean that a morphologically complex form cannot be analyzed using glosses in an alternate language without creating duplicate lexical entries, one for each metalanguage.

plementation, and user interface for configuring morphological parsers. OLD parsers consist of three components: a phonology that maps phonetic or orthographic representations to phonemic ones, a morphology that maps phonemic representations to licit sequences of morphemes, and a disambiguator which ranks morphemic analyses according to probability. OLD phonologies and morphologies are implemented as finite-state transducers and the disambiguators are built upon N -gram language models (cf. Jurafsky and Martin, 2008). OLD users can create any number of phonologies, morphologies, and disambiguators and combine them to create any number of parsers. Users specify phonologies as ordered context-sensitive (CS) rewrite rules using a notation and conceptual model with a long tradition of use in phonological research (cf. Chomsky and Halle, 1968). OLD morphologies are simply sets of morpheme category sequences that correspond to the category sequences of licit words; these can be specified manually or extracted automatically from a corpus created and/or specified by the user. Disambiguators are built upon language models extracted from a corpus specified by the user. OLD morphologies and disambiguators can be created without much dedicated effort on the part of the user, e.g., simply by creating a corpus of forms that, in their estimation, contain well analyzed word forms. Creating a phonology, however, requires a dedicated effort in order to formulate and order the requisite rewrite rules. The ability to create multiple distinct parsers within a single OLD application is useful in that it allows different users to experiment with different parsers that accord with their own analytical approaches; it also allows for the creation of parsers that take orthographic transcriptions as input as well as those that take

phonetic transcriptions as input. In addition, OLD parsers can be exported as stand-alone command-line utilities that users can use locally and incorporate into their own projects, if desired. Also, since OLD 1.0 applications are web services, parse functionality may be requested by a variety of applications. The OLD parser creation functionality is described in detail in chapter 3.

The FLEEx approach to morphological parsers is described in Black and Simons (2008) and the details of the computational implementation can be found in that work and its technical references. In contrast to the OLD approach where users create independent morphology and phonology components, the FLEEx approach to parser creation is more tightly integrated into the workflow of creating lexical entries. FLEEx users specify allomorphs for lexical entries, category inputs and outputs for derivational morpheme entries, and inflectional templates for the categories used to categorize entries. FLEEx assembles all of this information in order to generate the morphological parser for an application. In addition to the functional parser tool, the morphological information specified by FLEEx users can be used to automatically generate a human-readable sketch of the morphological component of the grammar.

Clearly there are benefits and drawbacks to each of these approaches. The OLD approach emphasizes the phonological mapping and permits only a relatively simplistic modelling of the morphology, i.e., with no explicit use of the concepts of derivation, inflection, or allomorphy. In contrast, the FLEEx approach encodes phonological mappings within allomorph specifications and allows for a more nuanced modelling of the morphology that

allows for the automatic generation of a morphological grammar sketch. In response to this, it should be pointed out that an OLD phonology consists of linguist-readable rewrite rules and may also contain comments that elaborate on the rules as well as tests representing mappings that the phonology should account for; OLD applications could, in the future, be made to transform this information (as well as the morphology information) into a more human-readable format that could constitute a grammar sketch. The ability to create multiple parsers (and parser components) and use them locally or via requests to an OLD web service is, in my estimation, a point in favour of the OLD parser implementation. As to performance comparisons, I cannot provide any; however, as Black and Simons (2008) point out, morphological parsers within the context of a fieldwork/documentation database application do not have high performance requirements since the objective is to parse words during user input and not large sets of existing words.

Butler and van Volkinburg (2007) reviews a version of FLEEx that comes with the FieldWorks suite version 4.0.1.⁵⁵ This review asserts that the system’s networking functionality—which allows for simultaneous multi-contributor access—is “fairly simple” to configure and use. It also praises FLEEx’s bulk editing capability and its features for creating dictionary representations. However, the reviewers have a lengthy list of complaints including crashes during concurrent access, inability to create reverse (i.e., gloss-to-

⁵⁵Butler and van Volkinburg (2007) lists the version reviewed as 4.0.1 yet Rogers (2010) claim to be reviewing version 3.0 and cite Butler and van Volkinburg (2007) as a review of an earlier version of the software. I do not know which source is inaccurate in this regard, but I assume that “4.0.1” refers to the version of the larger FieldWorks suite and not FLEEx itself.

2.2. Other fieldwork tools

vernacular⁵⁶) dictionaries, poor performance when editing interlinear texts, inability to search lexical entries according to the translation and notes fields, a very long wait for parser loading,⁵⁷ and difficulties in downloading the source code.⁵⁸ The authors also make the excellent observation that functionality for creating a gloss-to-vernacular dictionary creation would be very useful, especially to community-based revitalization projects.

Rogers (2010) is a review of FLEEx version 3.0. The author voices his appreciation for the following feature additions and improvements in response to the criticisms of Butler and van Volkinburg (2007) (some of which are mentioned above): functionality for creating dictionaries from the gloss language to the vernacular language, ability to specify that distinct lexical items are variants of one another, regular expression search across a variety of fields, and functionality allowing for syntactic labelling of the components of complex forms, i.e., the syntactic tagger. The reviewer complains of “sometimes . . . unbearably slow” interface response time (Rogers, 2010, p. 80), inability to access multiple views simultaneously, issues with importing IGT texts,⁵⁹ inadequate export formats (viz. no plain text export format), lack of features for annotating digital recordings, no capability for creating multilingual dictionaries, and no Mac version.

Clearly SIL International is a large organization with impressive resources and a large user base for developing effective fieldwork software

⁵⁶I would call this a dictionary from the metalanguage to the object language.

⁵⁷It is unclear to me whether this long wait time is for compiling the parser, i.e., modifying it, or if it is just for loading it for use. The former would make more sense and would be more understandable. The latter would be a more serious issue.

⁵⁸“Our developer gave up downloading the source code after it had run for 2 days!” (Butler and van Volkinburg, 2007)

⁵⁹The author was unable to import analyzed texts from Toolbox or any other program.

(see § 2.2.1). FLE_x improves upon Toolbox in a number of ways that parallel the features of the OLD as touted here—i.e., multi-user, network-based collaboration; multi-field, regular expression-included search functionality; and open source licensing—as well as in a number of ways not matched by the OLD—i.e., IGT text import, bulk editing capability, offline capability, and features facilitating dictionary creation, including the ability to create reverse dictionaries.

A primary advantage of the OLD over FLE_x is the more general, form-focused data structure of the former as contrasted with the more specific, lexical entry-focused data structure of the latter. FLE_x targets descriptive linguists whose primary goals are the creation of a dictionary, supporting texts, a descriptive grammar, and, ultimately (for many), a Bible translation into the vernacular. In FLE_x, lexical entries and sentences are completely distinct entities. However, in the OLD, morphemes, words, and sentences are all represented by the same type of object: the form. These types of form can be distinguished when necessary (by their category, tags, or by patterns in their morphological analyses) but they can also be treated as the same for the purposes of searching, embedding into documents, and building corpora. The OLD, with its support for grammaticality judgments and elicitation method categorizations, is currently tailored more towards theoretical linguists. However, support for dictionary creation can be seen in the (admittedly incipient) dictionary interface and the fact that the database can be used, as is, to amass and curate lexical items. Clearly there are opportunities to improve the OLD in this sphere, e.g., by updating the dictionary interface, allowing users to define custom lexicographic orderings,

and providing better support for form-to-form cross-referencing, e.g., for referencing example sentences, synonyms, related forms, etc. However, while such modification to the OLD would be additive, modifying FLE_x to be more OLD-like—e.g., so that morphemes, words, and sentences could all be queried simultaneously or referenced in research papers—would seem to require more foundational changes.

Arguably another advantage of the OLD is the fact that it is a web application and not, like FLE_x, a desktop application. While experience with Web 1.0 web sites and poorly constructed Web 2.0 “applications” may lead some readers to view this as a disadvantage, there are a number of reasons to think otherwise. In brief, these are a) a long history of support for multi-user concurrency, b) the benefits of the service-oriented architecture, c) platform agnosticism, d) the existence of high quality and rapidly evolving frameworks and libraries that allow for the creation of browser-based applications that are constructed according to road-tested design patterns, are thoroughly testable, and have graphical user interface features comparable to those of desktop applications. The first point means that building an application which allows multiple fieldworkers to collaborate on creating a single repository of language data is arguably easier using web technologies which have, since their inception, been required to adapt to multi-agent alteration of centralized resources. The second point alludes to the possibilities for the creation of software that creates new value by building upon pre-existing and independent web services, an example of which might be an application that aids in language learning by drawing on data provided by a number of web services that expose fieldwork-generated resources. The

2.2. *Other fieldwork tools*

third point refers to the fact that by housing application logic in web servers and client-side browsers, one can side-step the fractured platform/operating system environment that has long plagued software developers seeking to reach the broadest possible user base. The last point counters the conventional wisdom that web-based applications cannot match desktop-based ones in terms of being reliable, maintainable, testable, and usable. In addition, increasing interest in web application development means that developers' skills are, as a whole, moving in the direction of greater familiarity with web technologies and, therefore, the chances are greater that developers may be found when needed. A final advantage of the web-based approach results from the fact that web hosting services standardly have redundancies and backups that decrease the chances that valuable fieldwork data may be lost.

Though I have not used FLE_x in my own linguistic fieldwork, my research indicates that it is an excellent tool in a number of respects, as described above. Indeed, future development of the OLD will involve both borrowing certain features and ideas from FLE_x—e.g., aspects of its approach to bulk editing, its user-modifiable data structure, and its inclusion of word category and word gloss lines in interlinear analyses—while also improving import/export capabilities so as to enhance interoperability and complementarity with FLE_x. However, the OLD does, in its present state, respond to a real and present need for a fieldwork tool that is simultaneously general while catering to certain requirements of researcher linguists, focuses on web-based collaboration, and makes use of technologies and design patterns that facilitate the cooperative evolution of tools that advance fieldwork-related goals.

2.2.4 LingSync

LingSync⁶⁰ is “a free tool for creating and maintaining a shared database for communities, linguists and language learners” (LingSync, 2014a). It responds to the same needs as the OLD (cf. LingSync, 2013) and, as a result, is similar in many respects. Both tools seek to foster collaboration, data sharing, and data re-purposing among fieldworkers and other individuals and organizations with an interest in endangered language data. Like the OLD, LingSync is open source,⁶¹ web-based software that consists of a number of independent web services and client-side user interfaces. Also like the OLD, it is designed not specifically with lexicographic goals in mind (like Shoebox/Toolbox and FLEx) but with the broader goal of allowing fieldworkers to create general-purpose repositories of linguistic forms (cf. LingSync, 2013, p. 7).

While the OLD and LingSync have a lot in common, there are some salient differences. As is argued below, a foundational yet subtle conceptual difference consists in opposite rankings of the principle of collaboration relative to data privacy, rankings which help to explain some of the differences between the two systems at the feature level. Another major grouping of differences is largely technical and has to do with architecture, technologies used, and approaches to ensuring data privacy and ethical access. In brief, LingSync uses a No SQL (NoSQL) storage solution (as opposed to the SQL-interfaced, relational database back ends of OLD applications), is

⁶⁰<http://www.lingsync.org>.

⁶¹The source code for LingSync can be found at <https://github.com/OpenSourceFieldlinguistics/FieldDB>. It is released under the Apache License, Version 2.0.

2.2. Other fieldwork tools

coded almost entirely in JavaScript (in contrast to the Python/JavaScript logic of the OLD), and crucially employs encryption as part of its data access strategy.

This section begins with these conceptual and technical differences, discussing and evaluating them with reference to the OLD approach. It then moves on to a comparison of the two tools in terms of features, covering the advantages that each tool has over the other and discussing a few areas wherein both either excel or need work. The advantages of LingSync are, in brief, its flexible and user-customizable data structure, its use of encryption to provide improved data security, its functionality for making data public and discoverable and for automated transmission to institutionally-backed archives, its deployment approach which allows potential contributors to easily and immediately begin using the system, its activity feed, its import functionality, its offline capability, and its glosser module. The advantages of the OLD are its prescribed data structure which adheres to *de facto* standards and which is integrated into the application logic and interfaces, its columnar display of IGT data with visual feedback on lexical consistency of morphemic analyses, its text (i.e., collection) creation feature, its support for structurally searchable treebank corpora, its feature for creating bibliographies for source attributions of data, its orthography conversion and inventory-based input validation conveniences, and its functionality for creating morphological parsers and attendant implementation of morphological and phonological models. Both tools have partially overlapping yet distinct strengths in the following domains: software documentation, audio/video integration, data versioning, and search. Finally, both tools need to provide

better support for bulk editing and dictionary creation.

Both LingSync and the OLD could stand to benefit from borrowing and emulating certain features and approaches of the other. In fact, in line with the collaborative nature of the tools themselves, and given the extent to which we share common goals and approaches, I am currently engaged in a dialog with the group behind LingSync⁶² concerning collaborative development efforts,⁶³ in particular the creation of web services and GUIs that can interface with components of both systems, including morphological parsers, automatic annotation-audio aligners, and language learning applications.

Note that, like the OLD, LingSync is under active development and is adapting in response to the requirements of its users and ever-changing web technologies and web-based resources. Indeed, both tools are presently undergoing transitions such that the features discussed here may be spread across versions or components. The OLD is moving from a Web 1.0 application to a more modern Web 2.0⁶⁴ collection of tools consisting of a core web service and a single-page JavaScript application. LingSync is in the process

⁶²LingSync development has been, and continues to be, a collaborative effort between “students, professors, and software developers in the Montréal area, including: Alan Bale (Concordia, McGill), Gina Cook (iLanguage Lab, Concordia), Jessica Coon (McGill), Elise McClay (McGill), Gretchen McCulloch (McGill), Hisako Noguchi (Concordia), Tobin Skinner (iLanguage Lab, McGill)” (LingSync, 2014a), and others. The software currently has about 300 registered users and has been (or is being) used within a number of linguistic field methods courses offered by institutions including McGill University (Inuktitut), the University of Ottawa (Teenek), the University of Connecticut (Nepali), Yale University (Quechua), and Pomona College (Igikuria) (Gina Cook, p.c., LingSync (2014b)).

⁶³I have already made some small contributions to one of the LingSync client-side applications and is currently assisting with the development of the LingSync Spreadsheet GUI.

⁶⁴Note that “Web 2.0” refers not to the version number of the OLD but is web jargon that refers to a qualitative shift in the nature of web sites and applications over the past decade or so. According to Wikipedia, “Web 2.0 describes World Wide Web sites that use technology beyond the static pages of earlier Web sites”, cf. https://en.wikipedia.org/wiki/Web_2.0.

of re-implementing the features of its original online/offline Chrome app (hereafter dubbed “LingSync Prototype”⁶⁵) into “LingSync Spreadsheet,” a single-page JavaScript application written using the AngularJS framework that currently works online only. In addition, improvements to the core LingSync web service modules are ongoing.

2.2.4.1 Ranking privacy and collaboration

Both LingSync and the OLD seek to facilitate collaborative linguistic fieldwork while allowing contributors to keep their data private, as needed. Though both pieces of software share these primary goals and implement features towards their attainment, it is fair to say, in my judgment, that LingSync places relatively more emphasis on privacy (as opposed to collaboration) while the OLD places relatively more on collaboration (as opposed to privacy). This difference helps to contextualize and make understandable certain aspects of the feature sets of the two systems, as elaborated below. LingSync assumes that users will, individually or in highly coordinated groups, build private corpora and thereafter (if desired) grant access to other contributors or viewers, assuming that the new contributors will adhere to the conventions of the host corpus. The OLD, in contrast, assumes from the outset a state of affairs where multiple users contribute to a single, heterogeneously analyzed data set and rely on basic authentication and authorization plus the honour system to ensure ethical access and curation

⁶⁵A Chrome App is a piece of software written using web technologies (i.e., HTML5, CSS, and JavaScript) but which runs in the Chrome browser and, as a result, has additional capabilities, such as being able to run without an Internet connection and having access to a local file system, cf. http://developer.chrome.com/apps/about_apps.html

of data.

In addition to the obviously related care that LingSync takes toward ensuring data privacy via encryption, this basic difference in priority ranking helps to explain the differences in the ways that the two tools approach the lexicon and the conveniences surrounding it. The OLD anticipates the possibility that different users will (initially, at least) enter distinct and incompatible morphological analyses. The system therefore encourages the creation of lexical entries and supplies an interface that provides IGT-embedded feedback on the consistency of the morphological analyses with the extant lexical items. LingSync, in contrast, provides a lexicon module that automatically extracts the lexical items and morphotactic patterns implicit in users' analyses, assumes that these are relatively consistent, and uses these to provide the auto-glossing feature.⁶⁶

Another, admittedly relatively minor, area where this basic privacy-vs.-collaboration difference has an effect is in the rationalization of the *duplicate datum/form* feature implemented by both applications. In the LingSync documentation,⁶⁷ this is discussed as a feature for easily creating minimal pairs whereas in the OLD it is discussed as a feature that can allow a contributor to easily create an analytically re-analyzed version of another user's form without the complications inherent in modifying the original.

While this is a subtle distinction, understanding how the two pieces of

⁶⁶LingSync Prototype also provides a graphical visualization of the lexicon of a corpus as a network of nodes. Since nodes with few connections can be indicative of lexical outliers, one could argue that this feature is the functional equivalent of the OLD's morpho-lexical consistency feedback.

⁶⁷<https://www.youtube.com/watch?v=4Xr08AhGNqo&list=PLUrH6CNxFDrMtraL8hTLbLsQwdw1117FT>.

software prioritize privacy and collaboration relative to one another can help to understand the differences in design decisions and features (or lack thereof) discussed below.

2.2.4.2 Technical differences

This section discusses some technical differences between LingSync and the OLD. While it is relevant to LingSync's flexible data structure and encryption-based data protection feature, some readers may wish to skip ahead to the less acronym-filled sub-sections that follow.

LingSync data are stored as JSON objects within NoSQL databases: Apache CouchDB on the server and PouchDB⁶⁸ on the clients (cf. LingSync, 2013, 2014b). Since the data within these databases are stored as JSON-serialized JavaScript objects, LingSync avoids the performance costs inherent in converting relationally modelled OLD entities to Python instances and then to JSON objects. Moreover, since CouchDB and PouchDB are schema-less, users can add and remove attributes to the objects that encode their data points as they see fit; that is, these NoSQL storage technologies are designed with structural flexibility as a core feature. Perhaps the strongest argument for the use of CouchDB/PouchDB arises from the fact that these tools were expressly designed to facilitate synchronization between a central server-side database and client-side replicas; this means that offline access to data can be implemented atop the technology stack of LingSync more easily than atop that of the OLD. In fact, there are currently no tools that facilitate SQL-based access to relationally structured client-side data across

⁶⁸See <http://couchdb.apache.org> and <http://pouchdb.com>.

2.2. Other fieldwork tools

all browsers (cf. Lawson and Sharp, 2011). A final advantage of LingSync’s choice of database is that the data are stored in a human-readable format (i.e., JSON), as opposed to the binary data files of the RDBMSs⁶⁹ that manage OLD application data. This is desirable from an archival point of view. Of course, this criticism can be addressed by a) the proposed automatic publishing of OLD data sets to established archives and b) ensuring that XML database dumps are part of an OLD application’s regular backup procedure.

Note that browser support for size-unconstrained⁷⁰ persistent storage is currently fractured and appears to be at a standstill. The older Web SQL Database standard is supported by current versions of Chrome, Safari, and Opera but the World Wide Web Consortium (W3C) has ceased to maintain the specification⁷¹ and it is therefore likely that browsers will stop supporting it in the near future. The Indexed Database API (a.k.a. IndexedDB⁷²) standard is supported by current versions of Internet Explorer, Firefox and Chrome. However, there is no indication that it will be adopted by Safari or Opera any time soon. Tools like PouchDB abstract away from the underlying storage mechanism (in this case using Web SQL for Safari and

⁶⁹Note that the OLD has been tested with both MySQL and SQLite, though the latter RDBMS is not built for high levels of concurrency and thus should not be used in production. The database ORM abstraction layer, i.e., SQLAlchemy, allows for a range of other RDBMSs, including PostgreSQL and Firebird, both of which are open source. If needed, these could be used in OLD applications; though some minimal modification/parameterization would be required.

⁷⁰There is widespread browser support for the Web Storage (i.e., key/value pairs) standard and its cross-session persistence feature dubbed localStorage. However, the excessive size limitations of localStorage make it unsuitable for effective client-side persistence of a database of linguistic fieldwork.

⁷¹<http://www.w3.org/TR/webdatabase/>.

⁷²<http://www.w3.org/TR/IndexedDB>.

2.2. Other fieldwork tools

Opera, and IndexedDB elsewhere) in order to provide a uniform interface. However, there exist no comparable SQL-based abstractions.⁷³ Therefore, cross-browser storage of relational fieldwork data (e.g., OLD data) would require both re-designing the query logic (esp. search, cf. § 2.3.6) for the client and re-structuring the data as non-relational objects for client-side storage,⁷⁴ neither of which are very attractive propositions.

LingSync’s application logic, both on the server and on the client, is written entirely in JavaScript.⁷⁵ This approach has the benefit of freeing developers from switching between different programming language syntaxes and idioms and, in certain domains such as input validation, code can be reused on both server and client without the wasteful duplication that is sometimes necessitated by the Python/JavaScript server/client technology stack of the OLD. While JavaScript has been deservedly maligned—witness the impicature in the title *JavaScript: the good parts* (Crockford, 2008), one of the most widely referenced texts on the language—the explosion

⁷³Note that NoSQL databases do not expressly preclude *relationally structured* data, i.e., tables/objects referencing other tables/objects in order to represent one-to-many and many-to-many relationships, etc. However, they do forego implementation of the declarative SQL-based interface that is essentially a necessity for querying relational data.

⁷⁴Until, of course, some enterprising individual writes a JavaScript-based SQL engine atop the IndexedDB standard. However, this seems unlikely given that NoSQL solutions are very much in vogue at the present moment. In brief, the primary arguments for NoSQL databases point to their flexibility (i.e., schema-less-ness) and their ease of (horizontal) scalability, i.e., their ability to effectively handle extremely large data sets by distributing tasks across low-cost hardware. The advantages of relational databases include their maturity and familiarity, the conceptual elegance and lack of redundancy of normalized relational data, and the declarative language (i.e., SQL) for specifying complex queries, functionality leveraged to good effect in OLD search (cf. § 2.3.6). The article at <http://www.techrepublic.com/blog/10-things/10-things-you-should-know-about-nosql-databases> provides a well-balanced, and ultimately pro-NoSQL, overview of the debate.

⁷⁵The one exception to this is the LingSync Android application which is written in Java (cf. LingSync, 2013).

in browser-based application development over the past decade or so has resulted in a plethora of tools for creating (e.g., structuring, testing, etc.) sophisticated applications in the language.⁷⁶ The obvious response from the pro-Python developer would be to point to that language's relative maturity (and consequently larger catalogue of special-purpose libraries, including NLTK) and its arguably more readable (and hence maintainable) syntax. However, these considerations are, to a certain extent, subjective and, more importantly, largely irrelevant in the context of an ecosystem of fieldwork-facilitating web services that communicate via standard methods (HTTP and JSON) regardless of their underlying implementation.

2.2.4.3 Rigid versus flexible data structures

Both the OLD's prescriptive and system-integrated data structure and LingSync's flexible and user-customizable one have their benefits and drawbacks. These are reviewed here.

LingSync emphasizes its flexible model, or data structure, as a design feature. This dissertation, in contrast, argues for the utility of the OLD way of structuring linguistic fieldwork (cf. § 2.3.4), while recognizing that the system should be modified to allow users more freedom in modifying the data structure to their needs.⁷⁷ Clearly, fieldworkers will be more likely to use a tool that can be tailored to their pre-existing methods or ideas about how linguistic data should be structured. Since these tools seek to attract contri-

⁷⁶Note that the reason that JavaScript has received such attention from developers is that it is, for all practical purposes, the only programming language that can be run in a web browser.

⁷⁷As discussed elsewhere, future versions of the OLD will allow for a more user-configurable data structure.

butions from a broad spectrum of fieldworkers, such flexibility is undeniably desirable. On the other hand, there is a long tradition of scholarship in language documentation, description, and analysis from which a number of *de facto* standards have emerged. The various types of interlinearly glossed text formats are one set of examples, as are the lexical structures built into tools like FLEx and Toolbox, and the time-aligned annotation data structure of ELAN and similar tools. In order to make use of these conventions—e.g., to provide feedback on morpho-lexical consistency, to automate dictionary creation, to implement sophisticated input validation, to automate parsing and glossing, and to generate re-usable web-accessible data stores—the software needs to be aware of their forms and meanings. In short, creating a flexible and customizable model is one thing, but making *features* flexible to match is quite another. For practical reasons, therefore, some structural and semantic assumptions about the data will inevitably be embedded in the software.

In fact, LingSync does impose and assume particular structures for linguistic data. At the highest level there is the tree structure of corpora containing sessions containing datums.⁷⁸ This is, in fact, somewhat restrictive since not all data aggregated by fieldworkers comes from elicitation sessions—some may come from published texts such as research papers, grammars, or dictionaries. Contrast this with the OLD where corpora and elicitation sessions (a subtype of OLD collections) are both modelled as

⁷⁸As an aside, observe that since any registered LingSync user can create their own corpus and specify privacy settings, each user is effectively the administrator of their own corpus and, therefore, the system is arguably more democratic than the OLD where access settings for an entire language-specific database are controlled by administrators.

2.2. Other fieldwork tools

ordered lists of references (with repeats possible) to forms in the master repository.⁷⁹

Another in-built LingSync data structure is present at the level of individual datums, where the four fields *utterance*, *morphemes*, *glosses*, and *translation* are present by default and where their usage is necessary in order to provide automatic glossing functionality.

Also note that the flexibility of the LingSync data structure is limited by its two extant GUI applications, i.e., LingSync Prototype and LingSync Spreadsheet; in particular, hierarchical structures are not possible in either application. LingSync Spreadsheet offers a full template and a compact template, the first allowing six user-specifiable fields and the latter four. Here users may choose from a predefined set of input fields, i.e., column labels or object attribute names. LingSync Prototype (i.e., the Chrome app) offers more flexibility: users may add any number of fields to the interface, label them as they please, specify whether the data they contain are to be made confidential via encryption, and even provide metadata describing what the field should contain. However, neither interface allows for the creation of a hierarchical data structure, despite the fact that the underlying models, i.e., databases would permit it. That is, a LingSync user could not create a *source* attribute on datums comparable to OLD sources, i.e., one whose value is itself an object, i.e., a set of labelled fields, with attributes like *author* and *year*. Related to this, note that the value of the *tags* attribute of LingSync applications is a simple string, whereas in OLD applications a tag

⁷⁹Actually, it is only the Spreadsheet interface of LingSync that enforces the division of data points into disjoint session objects. The Prototype interface enforces no such requirement (Gina Cook, p.c.).

2.2. Other fieldwork tools

is an object in its own right, i.e., an entity with attributes such as *name* and *description*. This allows for more complex categorization of data and greater searchability. For example, OLD users can accurately search for forms associated to tags that contain a certain string in their description value, or for forms drawn from a source that was published between 2005 and 2010, etc. To do the equivalent in LingSync would require setting up conventions for encoding structured information as strings within field values and performing regular expressions to get the equivalent results, e.g., creating a *source* field in LingSync Prototype, populating it with values such as *Chomsky 1957* and *Sapir 1915*,⁸⁰ and finally performing the search by formulating a regular expression pattern like `/ 20(05|06|07|08|09|10)$/`.⁸¹

Note also that the fact that OLD tags, sources, categories, etc. are *bona fide* objects encourages consistency. That is, a user cannot tag something as, say, *generic aspect* without first creating a tag with that name. While some may view this as an unnecessary impediment to rapid tagging, the delay is, in my view, justified by the consistency it encourages. That is, a user is less likely to create a new, semantically redundant but formally distinct, tag like *aspect: generic* if they have to go through the process of intentionally creating that tag, a process which, as a matter of convention, should include

⁸⁰These values are oversimplified for exposition. To get the full equivalent of the OLD functionality would involve requiring users to compose JSON or some more simplified attribute-value serialization.

⁸¹Note that both FLEx and Toolbox offer greater flexibility than LingSync. In fact, Toolbox is arguably the most flexible since its users can create hierarchical data structures via the interface and can configure validation for the fields. FLEx allows users to specify custom fields/attributes for the hierarchically organized locations/objects that are built into the system (e.g., entries, senses, etc.); the OLD's move towards greater flexibility will probably follow the FLEx approach.

checking whether a relevant tag for their purposes exists already.⁸²

Of course, such considerations will not be novel to the clever developers behind LingSync. However, it should temper the temptation to appeal to an over-simplified rhetoric of “good” flexibility versus “bad” structure. The motivation behind the LingSync approach is clearly to allow users to configure their data structures to suit their needs and then later modify the application to make use of those structures once trends begin to emerge. In fact, this is already implemented to some extent in LingSync Prototype insofar as user-entered fields become visible by default in the interface once they pass a certain threshold of use. The point remains, however, that there is clearly a spectrum between unusably unstructured and smotheringly prescriptive with several positions along that spectrum, including the OLD’s, being defensible and having their own set of advantages, as described above.

2.2.4.4 LingSync advantages

Having discussed LingSync’s flexible and user-customizable data structure, this section reviews what are, in my opinion, the remaining primary advantages of that system. These are encryption-based data access control; functionality for making data public, discoverable, and for automatically transmitting them to archives; a low-barrier-to-entry deployment approach; the activity feed; import functionality; offline capability; and the automatic

⁸²This is really an argument for the relational model as much as it is for hierarchical structuring of data. That is, LingSync JSON objects with *source* attributes whose values are objects will contain a lot of redundancy when multiple datums reference the “same” source object. In order for LingSync to encourage consistency in the way being described, the application logic would need to aggregate past used source or tag objects and present them to users as suggestions.

glosser.

Encryption-based privacy. LingSync takes special measures to ensure that access to data can be effectively controlled. Users create corpora, which contain sessions, which contain datums, i.e., words, phrases, sentences, etc. The creator of a corpus specifies which (if any) other users are to be granted administrator, writer, or reader privileges to that corpus and encryption is used to control access. That is, data not specified as *public* are encrypted and can be decrypted only by authorized users. This ensures that even if confidential data are somehow leaked, they will not be decipherable by unauthorized individuals. This is an added security measure with clear advantages, and one that is not implemented by the OLD.

Publicization, discoverability, and archiving. One particularly valuable set of features implemented by LingSync allows users to tag subsets of their data as public, to make these data discoverable (e.g., via search engines), and to automate the communication of these data to language archives, e.g., the Open Language Archives Community (OLAC)⁸³ cf. LingSync (2013), thus further enhancing discoverability while also working towards long-term preservation of data. This is functionality that the OLD plans to implement but has not yet. Making data publicizable, discoverable, and archivable is, in fact, crucial to the overarching goal of making endangered and understudied language resources easily available to a wider audience (i.e., wider than the set of contributors to a corpus/database) so

⁸³<http://www.language-archives.org>

that research and revitalization efforts may profit.

Easy to start. An appealing feature of LingSync’s deployment approach is its publicly available *LingSync* server which allows users to get started quickly creating usable, shareable corpora. By way of contrast, in order to use the OLD to document a language for which there is currently no OLD application set up, one must download the software, install it on a server, purchase a domain, and have it resolve to the server’s IP address. Since this requires some level of technical proficiency and since it involves a delay in actually beginning to use the software for fieldwork, it is a considerable barrier to adoption. While LingSync’s public server is not (at least according to my understanding) meant to be a permanent solution for large-scale data collection projects (i.e., such projects will need to set up hosting for their own LingSync web services) it is nevertheless desirable as a means of allowing contributors to immediately begin experimenting with the tool.

Activity feed. LingSync Prototype’s activity feed—a widget that informs users of recent updates and other activity on the corpora that they are interested in—is a useful feature that the OLD could benefit from borrowing.⁸⁴ It is helpful for reminding a user of what they were working on previously or what other users have been doing in a shared corpus.

Import. LingSync (2013, p. 5) proposes the implementation of import from a number of formats, including “ELAN XML, CSV, and text file for-

⁸⁴Note that the activity feed is a feature present in the LingSync Prototype Chrome app (offline/online) but not currently implemented in the LingSync Spreadsheet web application.

2.2. Other fieldwork tools

mats.” As far as I have been able to discern, LingSync Prototype implements import from Comma-Separated Values (CSV) files while the LingSync Spreadsheet web application provides no import functionality as of yet. Since many software applications (e.g., FileMaker Pro, Microsoft Excel, LibreOffice Spreadsheet, etc.) allow users to export/save structured data in CSV format, this is a useful feature. LingSync Prototype provides an interactive import interface wherein users can view the data that they are about to import as a table and specify how to label the columns; that is, they can interactively specify how to map their structured data to the structure of the LingSync corpus that they are using and add additional fields/column labels as needed.⁸⁵ While LingSync’s import features are clearly still under development, the OLD, at present, unfortunately implements none whatsoever.⁸⁶

Offline. Offline functioning of full-featured fieldwork database applications is desirable. In my own experience, fieldwork involves either a speaker working with researchers in an institution with ubiquitous Internet access or a fieldworker travelling to a remote location with sporadic Internet access for a couple of weeks at most. Under these circumstances, offline capability is a desirable convenience but not really a necessity. Even in the latter scenario, well-functioning export and import capabilities are sufficient. That is, relevant data can be exported and consulted for in-field elicitation plan-

⁸⁵See <http://www.youtube.com/watch?v=Gcc0CFhGLfI> for a tutorial on how this works.

⁸⁶Though, again, since both tools expose an HTTP/JSON API, any competent programmer should have little trouble in writing a script to convert their structured data to JSON and then issue the appropriate POST requests to import their data to the central server-side database of either.

2.2. Other fieldwork tools

ning, data can be recorded in consistently structured formats, and then uploaded and imported to the system when web access is possible (usually in the evening at a motel). However, for many fieldworkers, fieldwork really is “in the field,” i.e., in remote locations where Internet connectivity may not be possible for several weeks. In such circumstances, inability to make use of the data and features of one’s web-based fieldwork software (esp. search) may significantly delay progress. Thus LingSync Prototype’s offline functionality⁸⁷ (as facilitated by its use of CouchDB and PouchDB as mentioned above) is a valuable feature that has no parallel (as yet) in the OLD.

Auto-glosser. LingSync implements an automatic glosser, i.e., a feature whereby the system suggests morpheme breaks and glosses to contributors during data entry. These suggestions are based on past analyses of the same words previously entered into the system. Soon to be implemented are variations on this feature, i.e., where the entering of a morpheme-segmented word will result in the auto-suggestion of glosses and transcriptions, or where user-entered morpheme glosses will serve as the trigger. This is a useful feature which has the potential to expedite data entry.⁸⁸

⁸⁷The LingSync Spreadsheet web app, does not, currently, function offline.

⁸⁸It is interesting to compare FLEEx’s morphological parsing features here. FLEEx suggests morphological analyses both a) simply by looking for exact matches of a word in its database and suggesting past user-supplied analyses of it and b) by using a true, user-configured morphological parser. These two types of suggestion are distinguished in the interface via colour-coding and the user can choose which, if any, to use (SIL International, 2014a).

2.2.4.5 OLD advantages

The advantage of a prescribed data structure integrated into application logic and interfaces has been argued for above and is also discussed extensively elsewhere in this dissertation, notably in § 2.3.4. Additional advantages of the OLD relative to LingSync include IGT-embedded feedback on lexical consistency of morphemic analyses, functionality for creating formatted texts with linguistic examples, structurally searchable treebanks, bibliography creation for source attribution, orthography conversion, inventory-based input validation, and functionality for creating morphological parsers and attendant implementation of morphological and phonological models.

Morpho-lexical consistency feedback. The graphical user interface of the OLD 0.2 displays forms in columnar interlinearly glossed text format by default and this presentation includes visual feedback on lexical consistency of morphological analyses. That is, users can immediately see a) which glosses correspond to which morpheme shapes (because words are aligned in columns) and b) the extent to which morphological analyses are consistent with the lexical entries already in the system. Experience has shown this latter feature to be much appreciated by groups of fieldworkers looking to ensure consistency of analysis.⁸⁹ The LingSync Spreadsheet interface

⁸⁹One issue with the OLD's morpho-lexical consistency feedback feature (pointed out to me by Lisa Matthewson, a reviewer and an administrator of an OLD application for Tlingit) is that since users can create their own incorrect lexical entries, the system can end up reinforcing multiple internally consistent but mutually inconsistent analysis patterns. To a certain extent, curtailing such behaviour is the responsibility of the administrators of a given OLD application. Nevertheless, a useful potential improvement to the OLD in response to this would be to have the system interactively alert users during data entry about possible lexical matches for the morphemes they are entering, perhaps even including usage statistics on said morphemes.

does not at present align words and their analyses into columns (though LingSync Prototype does) and neither of the two extant LingSync interfaces implement a morpho-lexical consistency feedback feature like that of the OLD.

That said, LingSync's auto-glosser feature plays a similar role to the OLD's morpho-lexical consistency feedback feature in that morphemes used in previously entered analyses will be suggested to LingSync contributors during data entry, thereby promoting consistency. In fact, since the LingSync auto-glosser presumably ranks its suggestions according to usage counts, it will be better at promoting overall consistency compared to the OLD approach which can actually promote inconsistency in the case where a rogue contributor continues to enter idiosyncratic analyses (i.e., analyses that are not consistent with those of the other contributors) and is reinforced in doing so by the visual feedback which indicates consistency with the lexical entries that he himself has created. One response to this critique is to point to the OLD's morphological parser functionality (cf. chapter 3) which is superior to the LingSync auto-glosser (since it can propose analyses for unattested words) and which will, when it is incorporated into the data entry GUI of the OLD, similarly promote overall consistency by suggesting a single most probable parse for a word. The other response is to assert that in some cases a system which allows for a plurality of analyses is in fact superior since at certain stages of research it is not clear which analysis should be preferred; a system that is constantly pestering the minority analyst to conform to the majority view may result in the mistaken abandonment of a superior analysis.

Formatted texts with linguistic examples. A feature not available in LingSync applications, and one which I argue to be highly useful, is the ability to create documents consisting of markup⁹⁰ and representations of data. These are the OLD *collection* objects, which allow users to create custom-formatted and exportable (to LaTeX) research papers and annotated representations of narratives, elicitation sessions, etc. out of the data present in the system. LingSync Prototype does allow for cross-session search results to be saved as exportable *data lists* (cf. LingSync, 2013); however, these do not allow for the interleaving of prose and markup with the data points. OLD collections are useful since they allow users to annotate sequences of forms and to create relatively sophisticated⁹¹ drafts of documents (e.g., research papers, annotated elicitation records, representations of narratives) containing their data.

Treebanks. Similar to the collections-as-texts discussed above are OLD corpora which are also ordered lists of references to forms. Corpora, how-

⁹⁰Currently, the lightweight markup languages Markdown and reStructuredText are supported. Both of these languages can be used to generate HTML; the latter can also generate LaTeX documents.

⁹¹Note that OLD collections can be associated to multiple files, i.e., images, videos, audio recordings, and/or text documents. I worked with a group on eliciting a traditional narrative with a Blackfoot consultant and we used this functionality as follows. The collection representing the narrative was itself associated to a) a long audio recording of the speaker delivering the narrative in a practised and uninterrupted manner and b) a video file consisting of still images depicting the content of the narrative with bilingual subtitles and the just-mentioned audio recording. The form objects that constitute the transcription and analysis of the narrative were then also each associated to a) smaller audio files of the speaker uttering the relevant form and b) image files that were drawn based on the story and used as a cue for the speaker during the elicitation of the uninterrupted recording. All of these media files are embedded within the HTML representation and can be viewed/played while reading the textual content of the collection. A further innovation on collections in the OLD 1.0, allows them to reference other collections, thus permitting users to create large, exportable texts from smaller, independent collection objects.

ever, contain no additional text (i.e., prose and markup) but are instead used a) to generate treebanks that can be searched for structural patterns using TGrep2 and b), in future versions, to generate other single-file representations that can be searched for cross-sentential patterns using, for example, NLTK's corpus utilities. The ability to search a fieldwork database for structural patterns is a great boon to search. LingSync does not currently allow for structural search of phrase structure treebanks⁹² nor does it (to my knowledge) permit cross-sentential (i.e., cross-datum) search.

Bibliographies. The OLD allows contributors to create *source* objects for citing the sources of data from published materials and for creating bibliographies. As of the OLD version 1.0, the data structure for sources is taken from the BibTeX specification and source modification requests are validated server-side according to that specification. Since an OLD form can reference a source, one can accurately and consistently indicate the origin of data that come from texts such as dictionaries or grammars. In future versions of the OLD, users will be given the option of auto-generating a bibliography for collections that contain forms with sources, and LaTeX exports will include the appropriately generated BibTeX bibliography file. Though relatively minor, this convenience further facilitates the creation of professionally formatted research papers and other texts from within an OLD application.

⁹²LingSync Prototype does currently provide a `syntacticTreeLatex` field by default, which the system assumes will contain valid phrase structures in the bracket notation conventions of the LaTeX tree-drawing package `qtrees`. The LaTeX export function then lists these tree representations under the (gb4e) IGT data representations. However, these trees can only be searched for structural patterns insofar as one can concoct elaborate regular expression patterns over them, i.e., not very far.

Orthography conversion. The OLD provides support for orthography conversion (a feature not implemented by LingSync). That is, administrators can specify simple mappings between orthographies so that users can create, read, update, and search transcriptions using a chosen orthography while the system transparently converts input to and from an administrator-specified, system-wide storage orthography. This allows a group of fieldworkers to create a consistently transcribed data set despite the fact that different users may be using different orthographies to transcribe the data. Since many endangered languages have only recently come to be written down, orthographic conventions are often still in the process of being established and it is not uncommon to have several competing writing systems, each with its own staunch adherents. Since OLD applications seek contributions from as wide a user base as possible, the orthography conversion feature can prove quite useful.

Inventory-based validation of transcription input. Administrators of OLD applications can also specify inventories of graphemes for the various built-in transcription attributes⁹³ of forms and configure input validation so that warnings are issued or errors returned when invalid transcriptions are entered. This inventory information is also used to generate field-specific keyboard widgets to ease data input when contributors do not have access to appropriate keyboard layouts for the language they are documenting.⁹⁴

⁹³The “transcription” attributes are *orthographic transcription*, *phonetic transcription*, *narrow phonetic transcription*, and *morpheme break*.

⁹⁴LingSync Prototype allows users to specify Unicode keyboards as well, though these are not based on field-specific character/grapheme inventories.

Parsers and model implementations. Two major features of the OLD that have no equivalents in LingSync are the morphological parser creator and the computational modelling of linguistic models that it entails. The OLD uses finite-state transducers (FSTs) N -gram LMs to allow users to create computational implementations of an unlimited number of phonologies, morphologies, disambiguators, and morphological parsers, cf. chapter 3. These can be used to a) provide feedback on the consistency of user-entered representations with user-specified grammatical models and b) provide automatic generation of morphological analyses and/or transcriptions, a boon to rapid data entry. LingSync does provide an automatic glossing feature which uses previously entered transcriptions/analyses to automatically suggest analyses and/or transcriptions during data entry/update. While useful, this approach is clearly not as effective as a full-blown parser which can analyze previously unseen forms and which can be used to empirically test phonological and morphological models.

2.2.4.6 Features present in both

Both LingSync and the OLD provide software documentation, support for integrating audio and video into their data sets, versioning of data points, and search. The following sections discuss the differences between the two systems as regards these features.

Documentation. LingSync and the OLD are fairly well documented given that both are open source projects under ongoing development. They have somewhat complementary strengths and weaknesses in terms of software

2.2. Other fieldwork tools

documentation. In general, LingSync has a lot of (primarily user-oriented) documentation resources which could stand to be integrated into a more coherent whole; on the other hand, the OLD has (arguably) better technical and conceptual documentation but needs to update its user guides. Note that both tools have contextual help information built into their graphical interfaces.

LingSync (2013) is a good reference for understanding the goals, projected high-level architecture, and *raison d'être* of the system. The best ways to discover what has actually been implemented (aside from experimenting with the software itself) and what is currently under development are to consult the developer's blog (LingSync, 2014b) and the milestones tracker on the LingSync's GitHub repository.⁹⁵ The slideshow tutorial for the LingSync Spreadsheet app⁹⁶ linked to from the main LingSync page (LingSync, 2014a) provides a quick overview of that tool and helps new users to get started. Finally, uploaded to YouTube are a large number of screencasts that can help new users to understand the system at a high level and to begin using both of the two current interfaces, i.e., LingSync Spreadsheet and LingSync Prototype. The YouTube playlist entitled *LingSync Tutorials*⁹⁷ lists most of them.

The OLD documentation consists of this dissertation, which provides a detailed description of and argument for the system; the documentation for the OLD version 1.0 (Dunham, 2013a), which explains how to download and

⁹⁵https://github.com/OpenSourceFieldlinguistics/FieldDB/issues/milestones?direction=asc&sort=due_date&state=closed.

⁹⁶<https://www.lingsync.org/#/tutorial>

⁹⁷<https://www.youtube.com/watch?v=4Xr08AhGNqo&list=PLUrH6CNxFDrMtraL8hTLbLsQwdw1117FT>.

install the OLD web service and includes a detailed specification of the data structure and RESTful HTTP/JSON API; and the OLD (0.2) User Guide,⁹⁸ which is included in each language-specific OLD 0.2 web application but which has not been updated for some time.

Audio & video support. Both LingSync and the OLD allow users to associate audio and video files to their datums/forms; once associated, these media files are embedded within the representations of the data and can be played while viewing the textual data. However, there are certain implementation differences between the two systems as regards this functionality. Highly relevant here is the mutual recognition of the value in facilitating the time-alignment of media files with transcriptions. Time-aligned transcriptions of audio/video recordings have the potential to be highly useful in the creation of tools that support language learning and revitalization.

The OLD currently allows for any form to be associated to any number of files, including audio and video files, and an audio file can be categorized as a recording of an object language, metalanguage, or mixed utterance. In addition, contributors can configure audio file objects to reference other such objects for their digital content while supplying start and end values to indicate sub-intervals of the parent audio file. Thus the OLD has limited support for time-aligned audio annotation insofar as a contributor may, using the mechanism just described, associate a form representing a transcription of an utterance to an audio recording of a speaker producing that utterance. Being able to create the sub-interval-referencing file objects just described,

⁹⁸See <http://www.onlinelinguisticdatabase.org/help/olduserguide>.

saves some time in that it obviates the need to actually splice and export subsections of larger audio files.

LingSync (2013) discusses both a “dream” module that would assist with automating the alignment process as well as functionality for importing ELAN XML documents. However, it is unclear whether or to what extent either of these components have been implemented. Whether or not it has yet been implemented, functionality for automating the alignment of transcriptions with media files will allow fieldworkers to greatly expedite the tedious task of manual alignment (and the slightly less tedious task of sub-interval start and end time specification facilitated by the OLD, as described above.) A relevant, audio-related convenience of the LingSync Spreadsheet application that is implemented is that which allows users to record audio directly into the browser. This could potentially save time.

Versioning. Both the OLD and LingSync implement versioning of data. That is, previous versions of forms/datums are retained. This allows users to restore data after deletions and modifications (if necessary), view the history of individual data points, and guard against malicious or accidental data corruption. The OLD accomplishes this by timestamping all forms and storing deleted and modified forms in a dedicated backup table.

Search. LingSync provides three distinct search interfaces: one integrated into LingSync Spreadsheet, a second in LingSync Prototype, and a third, web-based interface for searching one’s public corpora via ElasticSearch.⁹⁹

⁹⁹ElasticSearch, cf. <http://www.elasticsearch.org/overview/elasticsearch>, is a web service that provides search functionality for distributed data stores via a REST-

2.2. Other fieldwork tools

All three search interfaces allow users to search across all sessions within a corpus.¹⁰⁰ The ElasticSearch interface allows users to search across all corpora that are public and to which they have access. The LingSync Spreadsheet search functionality is quite basic and currently only allows for conjoined substring matches over all datum fields, i.e., no field-specific filters and no disjunction. The LingSync Prototype search functionality is more sophisticated in that it allows for regular expression patterns; any number of field-specific filters conjoined, disjoined, or negated (via !); and indicates matched patterns via highlighting; however, the search structure is flat, i.e., it does not allow for a hierarchy of filters with conjuncts/disjuncts at the non-terminal nodes.¹⁰¹

As described in § 2.3.6, the OLD exposes a RESTful interface for specifying (as JSON arrays) arbitrarily deep tree structure queries where non-terminal nodes are conjunctions, disjunctions, or negations. This interface automatically handles SQL joins, meaning that it is easy to search across the hierarchical structure of the OLD form objects themselves; e.g., one can search for forms associated to tags whose description values match the regex `/(Chomsky|Bloomfield)/`. In addition, the OLD auto-generates a se-

ful interface.

¹⁰⁰The LingSync Spreadsheet app also allows one to restrict a search to a single (elicitation) session.

¹⁰¹This is concluded after experimenting with the system. Note that LingSync Prototype provides a multi-field form interface for searching as well as a textual search field where users can specify complex queries using the syntax `fieldName:regex` to specify filters and where these filters can be conjoined via `AND` and/or `OR`. This latter, textual interface is more powerful since it allows for multiple filters targeting the same field/attribute. However, attempting to bracket sequences of coordinated filters crashes the query parser, i.e., the query `utterance:y OR (utterance:e AND utterance:x)` returns nothing despite the fact that, logically, it should return results. Removing the brackets results in a default right-branching structure, i.e., `utterance:y OR utterance:e AND utterance:x` is implicitly equivalent to `((utterance:y OR utterance:e) AND utterance:x)`.

rialized representation of the tripartite (i.e., shape/gloss/category) morphological analyses of forms, thus allowing searches to unambiguously target morphemes, a feat that cannot be accomplished via independent filters on morpheme, gloss, and/or category fields.¹⁰²

2.2.4.7 Features lacking in both

Two major areas where both LingSync and the OLD could stand to be improved are bulk editing and dictionary creation.

Bulk edit. LingSync and the OLD both need to provide better support for bulk editing, i.e., allowing users to perform multi-form/datum updates. In my experience working with fieldworkers, this is a feature that is continually requested. This is because groups of linguists doing research on a particular language are continually developing and changing conventions at the foundations of their analyses and often want to keep their databases in accordance with those conventions. As a result, manual curation of the data can be a time-consuming process, one which could be greatly expedited via an effective bulk editing interface. As mentioned above (i.e., in § 2.2.3), FLEx provides an eminently imitable implementation of this feature.

LingSync Prototype provides a simple command-line interface to the application’s sand-boxed file system (the “Power Users Backend”) which allows users to manipulate the data using JavaScript. Recently, the LingSync development team also announced the creation of a prototype “cleaning bot” that

¹⁰²The search interface just described is exposed by the OLD 1.0 web service and is not integrated into the GUI since there is no GUI. The OLD 0.2 GUI has a decent search interface, but does not possess the expressive power of the OLD 1.0 search interface.

was programmed to automatically propose corrections to the datums within a test corpus and then prompt the owner of the corpus to confirm/execute the proposed changes.¹⁰³ Similarly, the HTTP/JSON API exposed by the OLD 1.0 web service could be used to perform bulk updates by users who are capable of writing programs (in their language of choice) that can issue HTTP requests to retrieve the relevant data points, modify them accordingly, and then issue the appropriate update requests. In fact, the OLD source includes a simple module that hides the HTTP and JSON conversion details and would allow users to do this via the Python interactive prompt. However, despite these low-level solutions, the fact remains that a user-friendly interface that allows for configuring, previewing, and then applying bulk edits is still needed by both systems.

Dictionary creation. Both LingSync and the OLD provide only minimal support for creating dictionaries. LingSync’s automatically extracted lexica could be used as the foundation for generating dictionary-like resources (cf. LingSync, 2013, p. 22). The OLD 0.2, provides a dictionary-like interface (including customizable sorting) to the forms that it assumes to be morphemes as based on a problematically simplistic heuristic. However, neither system currently supports a data structure with the kinds of hierarchies and relations necessary for a rich dictionary representation, as is provided in software such as Toolbox and FLEx, e.g., specification of synonymy and variant relations, specification of distinct senses for a lexical entry, etc.¹⁰⁴

¹⁰³<https://www.facebook.com/LingSyncApp>.

¹⁰⁴Note that OLD forms can have any number of translations. While this does allow for the creation of lexical entry forms with multiple translations-as-senses, it does not allow for forms with multiple distinct senses where such senses are differentiated by dis-

2.2.5 Summary

This section has reviewed three fieldwork applications—Toolbox, FLEx, and LingSync—and compared them to the OLD. Toolbox has a long tradition of use in linguistic fieldwork and is well equipped to help with creating dictionaries and analyzing texts. FLEx has essentially the same feature set as Toolbox with various incremental improvements as well as novel features, notable among which is incipient support for multi-contributor network-based data creation. As a web application with a feature set that is general enough to be useful to both theoretical and descriptive linguists, LingSync is quite similar to the OLD; the discussion above explored the comparative benefits of each tool.

The reviews of Toolbox, FLEx, and LingSync provided above included direct comparisons between the features sets of those tools and the feature set of the OLD. In those reviews, efforts were made to argue for the superiority, or at least the equality, of the design and feature set of the OLD relative to those tools. However, the comparison also revealed a number of respects in which these other tools are superior to the OLD. This, in turn, suggests ways in which the OLD could be improved by borrowing features from these other tools or by increasing the potential for interoperability and complementarity between the OLD and these other systems. In the remainder of this summary, I itemize the relative demerits of the OLD and explain my strategy for addressing these in future work, either by implementing new features or by increasing the capacity for interoperability and complementarity

tinct morpheme break, morpheme gloss, and/or category values, something that is made possible by the data structures of Toolbox and FLEx.

between the OLD and these other tools.

Perhaps the primary demerit of the OLD is that there is no GUI for version 1.0. That is, there is no GUI that would allow one, for example, to exploit the powerful search functionality described in § 2.3.6, to build parsers using the morphological parser creator described in chapter 3, or to expedite data entry using the parsers so created. Creating such a GUI is the next major step in the planned development of the OLD. Additional improvements to the OLD as inspired by the reviews given above are as follows. The OLD would benefit from having a more flexible data structure, bulk update functionality, support for additional IGT rows (e.g., word glosses, word categories, and allomorphic morpheme break lines), a lower barrier to entry (i.e., by providing an OLD service that new users can simply sign up for, along the lines of LingSync), greater customizability in terms of how data are displayed (e.g., IGT display as well as a tabular display with movable and hidable columns), more morphology-relevant fields along the lines of FLEx (e.g., morpheme type and categorial combinatoric specification), more fine-grained data access control functionality (e.g., encryption), support for making subsets of OLD data sets public and easily archivable, real-time rendering of OLD texts (i.e., collections), and functionality whereby the system would offer to automatically create forms for morphemes and/or words that are implicit in analyses as they are entered.

As discussed above, the OLD has no import functionality and quite limited export functionality. Future development of the software will improve import/export capability so that users can more easily move data in and out of an OLD application in order to then, say, import it into FLEx (or some

other similar application) as a contribution to a dictionary creation project.

2.3 Features

This section details the features of the OLD that most contribute to its efficacy as a fieldwork tool. That it helps fieldworkers to share (and thereby create a useful repository of) language data is one of the most significant points in its favour (§ 2.3.1). Its high-level design approach, i.e., its architecture (§ 2.3.2), and its release under an open source license (§ 2.3.3) are crucial to the re-purposing of both the language data and the software's own functionality. Its data structure (§ 2.3.4), so it is argued, strikes an effective balance between shackling prescriptivism and laissez-faire structurelessness. While the graphical user interface of the older OLD web application has some valuable conveniences, the programmatic interface of the new OLD web service boasts a conceptual simplicity and accessibility that are crucial to achieving reusability of data and functionality (§ 2.3.5). Being able to quickly and accurately retrieve relevant data is crucial; as such, significant thought has gone into developing an effective search interface and additional functionalities that improve search (§ 2.3.6). Minor automations are presented in § 2.3.7, though discussion of the computational morphological parsing and the requisite model-implementing functionality are left for chapter 3. Several features contribute to achieving consistency (§ 2.3.8) in the data, thereby rendering them more searchable and more accessible generally. The dictionary-like interface to lexical items is presented in § 2.3.9, while § 2.3.10 discusses facilities for creating formatted documents containing rich

2.3. Features

representations of fieldwork data. Functionality for customizing restrictions on access (§ 2.3.11) to data is also discussed. Finally, the happy state of the software’s documentation (§ 2.3.12) is revealed.

As mentioned above, there are two versions of the OLD: version 0.2 which is at present being actively used for collaborative linguistic fieldwork and version 1.0 which is a web service that is awaiting development of a GUI before it can see active use. The features described in the sections that follow have been implemented¹⁰⁵ in either or both versions of the OLD. Table 2.3 summarizes which features are present in which application. The currently in-development GUI for the OLD 1.0 web service¹⁰⁶ will implement the OLD 0.2 features that are currently missing (e.g., orthography conversion and a dictionary interface) from the OLD 1.0 web service.

Feature	OLD 0.2	OLD 1.0
architecture	web application	web service
GUI	yes	no
API	no	yes
morphological parser creator	no	yes
corpus creation	no	yes
structural search	no	yes
inventory-based input validation	yes	yes
orthography conversion	yes	no
dictionary interface	yes	no
text (i.e., collection) creation	yes	yes
authentication & role-based authorization	yes	yes

Table 2.3: Features across OLD versions

¹⁰⁵Planned features that have not yet been implemented are explicitly described as such.

¹⁰⁶See <https://github.com/jrwdunham/dative>.

2.3.1 Sharing data

The core feature of the OLD is its facilitation of data-sharing between field-workers. I contend that easy access to peer data is, especially in the endangered language context, preferable (both individually and communally) to data sequestration. This section reviews the properties and sub-features of the OLD that conspire to produce the emergent macro-feature of data-sharing facilitation.

Linguistics is a science. Its object of study is the set of cognitive abilities that permit an individual to produce and understand a specified language. The raw data of linguistics are communicative signals (typically acoustic, but also visual), contexts wherein these signals may be used, and speaker judgments concerning the well-formedness of signals and their compatibility with specified contexts. On the basis of such data, linguists generate hypotheses about the knowledge that is encoded in the brain of an individual.

While much of empirical linguistics can be characterized as the recording of observed behaviour, it is possible (in certain domains more than others) to control variables and therefore to speak meaningfully of linguistic experimentation. Clearly this is so in a scenario where a phonetician formalizes a procedure for eliciting the production of a controlled set of tongue-twisters while making ultrasound image recordings of speakers' tongues. Though the variables are more difficult to control, the label of *experimental linguistics* could also arguably be applied to the semanticist who creates a representation of a contextualized communication event and collects speaker judgments on the acceptability of a particular utterance therein.

2.3. Features

Given that linguistics is a scientific endeavour, an obvious argument for empirical transparency is that it facilitates peer replication of experiments and observations, whence improved assessment of generalizations and claims. However, it is most often impossible for interested peers to replicate data collection procedures on endangered languages precisely because speakers of these languages are so rare. Moreover, the rapport and implicit set of understandings between researcher and language consultant often cannot be conveyed by representations of data. Therefore, while the detailed endangered language data provisioned by a system like the OLD might form the basis for questioning a researcher's conclusions and forming alternative accounts, convincing and publication-worthy counter-claims should generally be grounded in original empirical work.

The primary considerations supporting linguistic data-sharing are, in my estimation, the potential for fast-tracking incipient research and repurposing data toward other fieldwork-related endeavours, especially revitalization.

Having the opportunity to browse and effectively search peer data is a great boon to research, especially in its beginning stages. Given such an opportunity, imagine how much more quickly one could uncover, say, minimal pairs, phonotactic patterns, possible permutations of verbal inflectional morphology, or the distribution and semantic contribution of a particular morpheme. During the discussions of research groups or those that arise after presentations, it very often happens that a line of thought will become stalled on an empirical point that could quickly and easily be settled by access to a communally generated repository of language data such as an

OLD application.

The other primary positive result of widespread sharing of endangered language data is the potential for reuse beyond research. I am thinking specifically here of the potential for using such data in the teaching and learning of the languages under consideration. In the course of a two-semester linguistic field methods course at UBC, I estimate that hundreds of hours of audio recordings and thousands of lines of transcriptions, translations and morphological analyses are generated. If these data are consolidated, structured, and easy to access, then they can be used to generate lesson plans, study exercises, and grammars and fed into dynamic language-learning software.

An OLD web application facilitates the sharing of language data among fieldworkers by providing an interface to a centralized database that can be modified by multiple contributors simultaneously. The degree to which data are effectively shared depends on the quality of the database schema and the application interface, i.e., how well the data are structured and presented, how consistent and detailed they are, how quickly and accurately they can be accessed (e.g., via search), and, generally, how enticing the user experience is. In short, the effectiveness of the core feature of data-sharing depends upon the effectiveness of the various features described and evaluated in the sections below.

Since some data-sharing-relevant features are too minor to deserve their own sections, I summarize the argument here. The database schema (i.e., structure) of the OLD is grounded in my own fieldwork experience with modifications resulting from the usage and feedback of dozens of fellow fieldwork-

2.3. Features

ers and OLD contributors. Data are presented cleanly (i.e., without clutter and with secondary¹⁰⁷ values hidden by default) and in the familiar interlinear glossed text format that is the *de facto* standard for foreign language data in research papers. Powerful search functionality allows for highly nuanced queries that may make use of regular expressions and may reference multiple attribute values, including (system-generated) morphological representations. Values for enterer and for date and time of entry are auto-generated to facilitate proper attribution. For the prevention of data loss and/or corruption, there are features that prohibit deletion of another contributor's forms and the behind-the-scenes preservation of past data states upon each successful modification. Data entry, modification and search are facilitated by conveniences of the user interface which include auto-fill of likely-to-be-repeated past values, keyboard shortcuts, and tab-based navigation of fields. Contributors can also collaboratively create exportable documents containing system data using the collection objects.

Before closing this section on the data-sharing feature of the OLD, I would like to address two related and valid concerns about the very desirability of such a feature. These are the reluctance of researchers to share their data and the requirements of speakers and communities that certain data not be made public. The OLD's features for controlling access to data, as discussed in § 2.3.11, are designed specifically to address concerns of this nature. However, in an environment where there is competition for publications, grants, and jobs and where research is based upon a rare and hard-

¹⁰⁷I informally divide attributes of linguistic forms into primary and secondary categories, with the former including the transcriptions, the morpheme break, the morpheme gloss, and the translations and the latter including everything else.

2.3. Features

won resource, researchers may be understandably hesitant to expose a fully candid data set. As it stands, it is possible for a registered OLD user to contribute data for a short time (or not at all) and then have access to all of the data in the system that are contributed by others. While the OLD does not prescribe particular terms of use for language-specific OLD applications, the administrators of such applications are encouraged to adopt a set of terms and require their acceptance prior to user registration. These terms should minimally specify whether and which data can be used in publications (or commercial products) and how the authors of such data are to be cited. If a user takes data from the system and uses it to further their own goals (without proper attribution or offers of co-authorship) and is conspicuously lacking in their own contributions to the system, then administrators may decide to close their account. Which is to say that, at this point, the OLD has no formal mechanisms in place for dealing with disputes of this nature. At this point, I am undecided about whether addressing this type of issue is a requirement of the OLD *per se*.¹⁰⁸ In my experience, fieldworkers can be trusted to behave ethically with respect to using the data of their peers. In addition, the set of research topics is so vast compared to the available researcher time that directly competing with another fieldworker for research results using that fieldworker's own data is an unlikely circumstance. For those still unconvinced, it hardly needs stating that a fully candid data set is not required: culturally sensitive and competitively vital data points

¹⁰⁸The approach taken by LingSync is arguably superior in this regard. In that system users control access to their own corpora and may allow (or revoke) different levels of access to specific users. The disadvantage of this, of course, is that there is, as a result, something of a barrier to the ideal of maximizing the sharing and the reuse of data.

may be withheld from the system according to the best judgment of the researcher and/or speaker/community. That said, further improvements to the authorization system of the OLD may be required in order to address lingering concerns of this nature.

However, the real challenge in getting fieldworkers to share their data is, in my estimation, enticing them to do so by creating a tool that complements their preferred elicitation and organizational practises and simultaneously adds real value to the process, e.g., by facilitating more rapid data creation via tools that, say, automate morphological parsing. My contention, as argued below, is that the OLD does just this.

2.3.2 Architecture

The architecture of the OLD—i.e., its high-level design (cf. McConnell, 2004)—is a feature in its own right, and one that is significant enough to the accomplishment of fieldwork goals that it deserves some discussion here.

The OLD 1.0 is a web service. This means that it exposes a simple yet powerful web-based interface that allows it to be used by other software. Any program capable of generating and sending HTTP¹⁰⁹ requests and receiving and parsing HTTP responses can interact with an OLD application (assuming, of course, that said program has a valid username and password authorizing the request issued.) Requests are typically one of the five denoted by the CRUD acronym, i.e., requests to search (S), create (C), read (R), update (U), or destroy (D) a resource (cf. Martin, 1983), i.e., an OLD

¹⁰⁹HTTP is the standardized communication protocol of the World Wide Web (Fielding et al., 1999). It is, in essence, simply a specification for how clients should format requests and servers should format responses.

2.3. Features

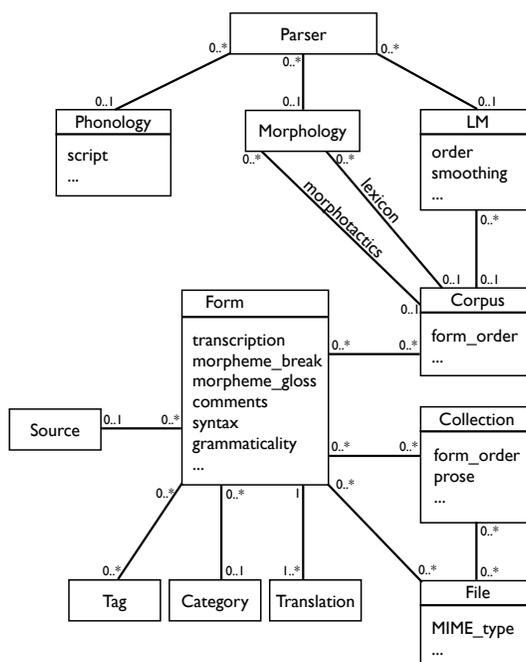


Figure 2.2: OLD UML diagram.

form, file, collection, user, category, parser, etc.

Figure 2.2 is a Unified Modelling Language (UML) diagram showing the main components of the OLD. The boxes represent classes and the lines connecting them represent “has a” relationships between the classes, with multiplicity indicated at the ends of the lines. Thus Figure 2.2 shows that an OLD form has zero or one sources ($0..1$) and a source has zero or more forms ($0..*$). The figure also shows that a form has one or more translations ($1..*$) and that each translation has exactly one form (1).

To get all forms within an OLD application being hosted at [http:](http://)

2.3. Features

`//my-language-specific-old.com`, one would issue an HTTP GET request on the URL `http://my-language-specific-old.com/forms`. This request would return an HTTP response whose body would contain a JSON array of objects, each representing one of the forms in the database. In order to create a new form on this OLD application, one would issue an HTTP POST request on the URL `http://my-language-specific-old.com/forms` with a JSON object in the request body that contained the attributes and values of the form to be created. The updating, deleting, and searching of forms are operations that are accomplished using similar HTTP request patterns. Files, collections, corpora, categories, phonologies, morphologies, language models, parsers, etc. are all created in an analogous fashion. For a comprehensive account of the data structure and API of the OLD 1.0, see the documentation at `http://online-linguistic-database.readthedocs.org/`.

An especially obvious type of software that could interact with an OLD 1.0 web service is one designed for user interaction. Such an application is currently under development.¹¹⁰ It runs client-side in the browser and is written in JavaScript, HTML, and CSS. The past decade has seen a dramatic increase in the capabilities of browser-based applications. In particular, now possible are interfaces completely lacking in page refreshes (cf. Galli et al., 2003) and possessing functionality such as drag-and-drop. It is also possible to create applications that run in the browser even when there is no Internet connection and which can store data indefinitely in browser-internal databases (cf. Lawson and Sharp, 2011). A plethora of libraries and frameworks have arisen to facilitate employment of these capabilities.

¹¹⁰See `https://github.com/jrwdunham/dative`.

The browser-based interface to the OLD is written in CoffeeScript (a language that compiles to JavaScript) and uses the Backbone framework (to structure the code) and the jQuery library (to facilitate the interface construction). Once completed, it will make use of these new capabilities in its re-implementation of the existing OLD 0.2 functionality while adding interfaces for the newly introduced resources (e.g., corpora and parsers), some level of offline capability, and a generally improved user experience.

Currently included with the source code of the OLD 1.0 is a simple Python module that uses the Requests library to interact via HTTP with a live OLD application. This module could serve as the basis for a more sophisticated Python desktop application, either one with a graphical user interface or one that resides in the command-line.

An interesting possibility opened up by the modularized web service architecture of the OLD 1.0 is an interface to multiple OLD applications. This would allow researchers to perform searches across a number of languages and compose drafts of research papers with arguments grounded in the cross-linguistic examples found.¹¹¹ I hope to implement this type of application at some future time.

Other enticing possibilities opened up by the OLD web service include independent applications such as audio dictionaries and general-purpose language learning tools which harvest structured multimedia data from OLD applications.

The architectural decision to design the OLD as a re-purposable web service allows for all of these different types of application to make use

¹¹¹Thanks to Martina Wiltschko for this interesting idea.

of it. It is thus a feature with significant potential for contributing to the accomplishment of fieldwork tasks.

2.3.3 Open source & web-based

Since the OLD is open source, it is freely available and its source code is accessible for inspection, contribution, and/or derivation. Being web-based, it works across operating system boundaries and has a lower barrier to developer entry when compared to software written in more esoteric or low-level programming languages. These properties make it easy for fieldworkers to use the software for a variety of purposes.

The OLD 1.0 is licensed under the Apache License, Version 2.0. This means that its source code is freely available for reuse. The source code may be used in whole or in part in the creation of other software, so long as the notice of the Apache License, Version 2.0 and notice of copyright holder are maintained unaltered within the source. Interested parties can browse and download the source at the OLD's GitHub repository. Using GitHub and Git, developers can contribute improvements to the software or clone it and begin developing a derivative work of their own.

As open source software, the OLD is more than just free of charge; it is freely available for detailed inspection and modification. This means that developers may study the source in detail in order to understand how it works, improve it, or even create new pieces of software that make use of it or parts of it. Because of this openness, the OLD is both a ready-to-use fieldwork tool and a resource of information and reusable components that may prove beneficial in the larger arena of what may be termed *computa-*

tional fieldwork. The present dissertation and the extensive documentation included in the source (see § 2.3.12) enhance this potential for reuse.

Since the OLD is web-based software, it is not subject to the cross-platform compatibility issues that can plague desktop-based applications. All modern computing platforms provide access to the World Wide Web via a web browser application and OLD applications can, therefore, be used on any platform.

Being web-based further increases the potential for contributing to and making use of the OLD code base. This is because the technologies employed—i.e., Python, HTML, CSS, and JavaScript—are widely used and relatively easy to learn. In my experience, most programmers (and many non-programmers) have some level of proficiency with the core web technologies, i.e., HTML, CSS, and JavaScript.¹¹² Python, the language in which the OLD server-side logic is written, is a mature, high-level programming language that is easy to learn and designed to be highly readable (see Lutz, 2013).

2.3.4 Data structure

The data structure of the OLD is the way that the system organizes the data it stores. The data in an OLD application are stored in tables in a relational database. However, the structure is here described using the more

¹¹²HTML is used to create web pages and is essentially a set of tags for organizing text into nested structures such as divisions, headers, paragraphs, and bullet lists. CSS, i.e., Cascading Style Sheets, is the language for defining how these structural elements are to be presented, e.g., the font size of headers, the background colour of containers, etc. JavaScript, the programming language of the web browser, allows for the creation of client-side application logic, e.g., changing a web interface in response to user actions without requesting new HTML pages from the server.

2.3. Features

transparent language of objects and their attributes. Appendix A explains relational databases (what they are, how they work, and the way complex objects are encoded within them) and details the OLD's data structure. The present section considers some of the more foundational and otherwise notable features of the OLD data structure and argues for their utility in accomplishing linguistic fieldwork goals.

The OLD data structure is an implicit prescription for how fieldwork data ought to be organized. This structure is based on my own fieldwork experience and has been refined and modified based on feedback from users of the nine language-specific OLD web applications currently in use.¹¹³ My contention is that the data structure described here is one of several possible such structures that would be effective for organizing fieldwork data. As I am continually refining the structure, it would be illogical to assert that it is ideal.¹¹⁴ However, it embodies some interesting design decisions that are crucial to other features and which, on the whole, contribute to the effective accomplishment of fieldwork tasks.

The first thing to consider is the value of giving any structure at all to language data. After all, many fieldworkers get along just fine by keeping

¹¹³The conventions for IGT representations in the OLD conform, for the most part, to the Leipzig Glossing Rules (cf. Bickel et al., 2008). However, note that, at present, the OLD is not set up to recognize reduplication via the Leipzig standards.

¹¹⁴I fully expect that the structures used by individual fieldworkers in their desktop database applications will be distinct from that described here. However, experience suggests that migrating data from a particular fieldworker's idiosyncratic data structure to that of the OLD is usually possible and often relatively painless. The more difficult cases necessitate writing scripts to pre-process data prior to migration. In some cases the OLD data structure may lack appropriate objects or attributes to accommodate some data type employed by a particular data set. As the data import functionality of the system evolves, it will probably be endowed with a data structure that can, to an extent, be modified in order to handle such cases.

2.3. Features

handwritten journals or creating a series of digital text documents. The case for structured data rests on the fact that structure facilitates consistency and more efficient access to data. In particular, structured data can be searched with greater accuracy and they can be manipulated programmatically more easily.

Setting aside hand-written notes, imagine the difficulty in searching through digital text documents for data containing a particular string that indicates the presence of a certain morpheme. Even assuming that searching across multiple documents is not a problem, it will be exceedingly difficult to restrict results to data where the target string is present only in the morphophonemic portions of the data and not in, say, the orthographic transcriptions, translations or free-form notes. Depending on the consistency with which the fieldworker has informally structured their data-containing documents, it may be possible to compose regular expressions that are up to the task; however, this will be very time-consuming and already presupposes a rather sophisticated technical expertise.

When linguistic fieldwork data are structured, i.e., when they are divided into logical sub-parts and labelled, it then becomes much easier to search them quickly and accurately. Consider a database table full of sentences with orthographic transcriptions, morphologically segmented phonemic transcriptions, glosses, translations, comments, and categories, all in their own labelled columns. Given such a table of structured data, a database management system can be used to perform searches based on multiple conditions, each requiring that different parts (i.e., columns) of the data have certain properties.

2.3. Features

Structured data can also be manipulated by computer programs far more easily than unstructured. This opens up possibilities such as programmatically extracting morpheme and word lists and other targeted sub-parts of the data for use in research or in the creation of pedagogical materials.

The OLD proposes four core objects for structuring linguistic fieldwork data: form, file, collection, and corpus. I contend that this represents a sound method of organization which accords with existing practises while also laying the groundwork for the fieldwork-enhancing features provisioned by the OLD.

Forms are textual representations of morphemes, words, phrases or sentences. The cap at sentences is a matter of convention—the hard restriction is actually a rather arbitrary 255 character limit on transcription values (which could be increased if necessary). The vast majority of linguistic examples in research papers, descriptive grammars, dictionaries, and learning materials can be represented by OLD forms. An obvious potential demerit of this approach is the fact that it is not possible to represent a multi-sentential datum as a single form. That is, in some instances it would be useful to be able to search for patterns that cross sentence boundaries. To a certain extent, this is made possible by the corpus construct which allows for the searching of representations of sequences of forms. If users need simply to create presentational representations of multi-sentential data, then collection objects can be used.

Another potential issue with form objects is that they conflate abstract lexical items with representations of records of speech events. That is, a bound morpheme (never utterable in isolation) and a sentence spoken by

2.3. Features

a particular speaker at a particular time are represented by the same data type. As a result, assigning values to certain attributes does not make sense for certain form types. For example, a bound morpheme should not have values specified for elicitor, speaker, and date of elicitation. This information should be encoded in the forms representing records of speech events in which the abstract morpheme is used. Although it does not currently, the interface to the data could easily be made to dynamically retrieve such substantive examples evincing the abstract forms, i.e., a user action on a lexical item would bring up a list of forms containing that lexical item. Forms representing bound lexical items should probably also not have phonetic transcriptions for the very reason that they are not uttered in isolation and therefore phonetic representations do not make sense for them.

Contributors need to be mindful of the abstract/substantive distinction when making decisions as just mentioned as well as when deciding whether to create duplicate entries. Abstract forms should not be duplicated unless the duplication can be motivated by significant novelty in analysis. On the other hand, the duplication of substantive forms may be more easily justified since differences in speaker, dialect, and context of use may be judged sufficient.

In practise, I have found that the benefits of being able to search across abstract and substantive forms simultaneously outweighs any costs arising from the conceptual vagueness of this conflation. When distinctions must be made, they can be made by filtering forms according to the relevant attribute values, e.g., by restricting search results to only those forms that have the categories corresponding to abstract lexical items, whatever those may be, given the analyses of the contributors to the application.

2.3. Features

Other notable design decisions of the form object include the grammaticality attribute, the possibility of multiple translations (with their own appropriateness specifications), morphological representations (and the conventions assumed therein), the distinction between general and speaker comments, the multipurpose tagging system, the elicitation method attribute, and the category attribute.

Linguists find value in data that indicate constructions that are *not* considered well-formed by speakers. For example, post-positions are not present in English so speakers will judge (3) to be malformed.

(3) *He went the store to.

There are conventions for representing various types and degrees of malformedness, with the asterisk as an indicator of ungrammaticality (cf. 3) being the most recognizable. The OLD data structure encodes grammaticality as a distinct forced-choice attribute. Administrators of the application can specify the available grammaticality options. This approach ensures that forms can easily be sorted by grammaticality without relying on searches of potentially inconsistent grammaticality representations in the values of various transcription attributes.

There is no limit on the number of translations that a form may have. The decision to use distinct translation objects as opposed to a single translation field means that users of the system do not need to deal with inconsistencies in how distinct translations are delimited, e.g., via informal conventions such as commas or semicolons. In addition, each translation object has its own appropriateness attribute which allows contributors to indicate, in a

consistent manner, whether a particular translation is compatible with a given form.¹¹⁵

As is discussed more thoroughly in the chapter on morphological parsing, the OLD assumes for certain functionality that morphological analyses are represented as sequences of morphemes with words separated by spaces and morphemes separated by delimiter characters (typically “-” to represent affix boundaries and “=” for clitic boundaries, (cf. Bickel et al., 2008)) that are specified by administrators in the application-wide system settings. Based on this assumption, an OLD application will attempt to identify morpheme shapes and glosses in the morpheme break and morpheme gloss fields and then attempt to match these against lexical items present in the database. Using any matches found, the system will generate values for additional attributes (e.g., category string) and will create representations of forms which make it clear to what extent the user-supplied morphological analysis is compatible with the lexical entries present in the system. In order for this to work, contributors must follow certain conventions, such as ensuring that there are neither space characters within individual glosses nor unintended delimiters within morpheme shape representations. By and large, this approach accords with the representations of morphological analyses used by linguists. It also reflects the standard analytical assumption of a hierarchy of representations wherein sequences of phonemically transcribed morphemes are mapped via phonological transformation rules to phonetic

¹¹⁵Note that this *appropriateness* attribute of translation objects is given the label *grammaticality* in the underlying data structure. However, since this does not really (typically) indicate the grammaticality of a translation, the label is not entirely accurate. Hence the use of “appropriateness” here.

2.3. Features

representations.

However, in many traditions an unsegmented transcription is not supplied since this would be highly redundant given the highly transparent morphophonology of the languages in question. An example is furnished by Salishan linguistics where an unsegmented transcription is not generally provided since the phonological transformations are simple enough that they can be supplied by the reader.¹¹⁶

The general comments attribute of forms is meant as a catch-all for fieldworker notes on the form in question. Contributors may adopt certain easily searchable conventions in the textual value of the comments field in order to categorize forms according to their purposes. However, the use of tag objects is the recommended approach for such categorization. Tags are user-created objects with names and descriptions; any number of tags may be associated to a given form. Reifying tags in this way avoids inadvertent, notational inconsistencies and facilitates the sorting of forms according to the categorizations implied by the tags. Quotations or paraphrases of the speaker on a particular form should be placed in the value of the speaker comments attribute. This type of comment is conceptually distinct enough from general comments that it deserves its own attribute. Often it is very insightful to approach linguistic data by focusing on the speaker's own in-

¹¹⁶The OLD should be able to accommodate this approach by allowing the morpheme break value to count as the required minimal transcription value. Currently the system requires an *orthographic* transcription for all forms but this is a restriction that should be lifted in favour of a more flexible one that allows any of the various transcription types to meet the minimal one-transcription requirement. Since orthographic transcriptions are desirable to the broader community of fieldworkers and language communities, the system could be configured to auto-generate delimiter-less transcriptions (either upon data entry or upon export.)

2.3. Features

tutions about them and a dedicated speaker comments attribute facilitates this.

A dedicated forced-choice field for morpho-syntactic categories is also well-founded. This information helps with sorting forms into abstract lexical items vs. records of utterance events. Of course, the system must still be told, in some sense, that forms associated to a tag with a name value of *Agr*, for example, are abstractions while those categorized via an *S*-labelled tag are not. The category information supplied by this attribute is also crucial to the system-generated syntactic category string and break-gloss-category values, as described above.

The elicitation method object used to classify forms is also notable. This information can be useful when assessing whether a datum is solid evidence for a particular claim. In semantic research, for instance, data elicited by requesting translations of metalanguage forms may not, on their own, be sufficient as evidence for certain claims. Data elicited by describing a carefully controlled context and, for example, requesting that the language consultant judge whether a provided form is felicitous within that context are another valuable type of evidence. In order that the relevance of data points can be assessed by researchers on their own terms, these different methods of elicitation should be recorded via the elicitation method attribute¹¹⁷ and any information about the context provided can be included in the value of the general comments attribute.¹¹⁸

¹¹⁷It may be useful to specify defaults for the elicitation method values. I invite suggestions for best practices in how to best categorize elicitation methods.

¹¹⁸The OLD data structure now includes a *context* field on form objects for the specification of such contextual semantic information.

2.3. Features

Text is not, of course, the only medium of language data. Audio and video recordings of speech are valuable and oft-created artifacts and these are encoded in OLD applications as file objects. The OLD allows for multiple files to be associated to a single form, reflecting the fact that multiple digital files may be relevant to a given form. Searches across forms can also contain conditions on the file or files to which the forms are associated.

The OLD currently supports some degree of indication of the nature of the relation between a form and an associated file. At present, one can specify, via the utterance type attribute, whether an audio file is a recording of an utterance in the object language, one in the metalanguage,¹¹⁹ or one that is mixed (i.e., both object and meta). Programs that make use of OLD data may use these specifications and assume that when a form is associated to an audio file classified as an object language utterance that the file is a recording of an utterance of that very form.¹²⁰ Such programs could then use this information in the creation of, say, learning games that get players to recognize spoken data.

The data structure for OLD files allows for three types of audio file: those whose digital content is stored on the OLD application server, those whose content is served elsewhere, and those whose content is a sub-portion of another audio file. The last of these is constructed by specifying a parent

¹¹⁹A recording of an utterance in the metalanguage could be useful for non-visual language learning tools. A simple example would be a series of ordered utterance-translation audio file pairs that a learner could listen to on an MP3 player in order to improve their vocabulary.

¹²⁰Of course, this assumption is problematic since a form may be associated to an object language audio file that *contains*, but does not *consist in*, an utterance of the form. For this reason, the OLD data structure will soon be updated so that the very *associations* between forms and files are what are categorized.

audio file and start and end times within the recording; these are useful for expediting the identification of smaller audio clips that correspond directly to forms, without performing the time-consuming task of manually editing audio files.

The data structure also encodes corpus, collection, phonology, morphology, morphological parser, and morpheme language model objects. These are discussed in detail in various sections below. There are also backup objects created for collections, corpora, and forms whenever one of these is updated or deleted. This helps to mitigate the danger of inadvertent data loss.

This concludes the discussion of the data structure of the OLD. While it will continue to evolve in response to user needs, it currently embodies a number of design principles that are original and which contribute to the effective accomplishment of fieldwork goals.

2.3.5 Interface

The various interfaces of the OLD contribute to its effectiveness as a fieldwork tool. There are in fact three interfaces: the OLD 0.2 GUI, the in-development OLD 1.0 GUI, and OLD 1.0 application programming interface (API).

2.3.5.1 OLD 0.2 GUI

The OLD 0.2 GUI boasts an uncluttered design, keyboard shortcuts to expedite navigation, and integrated documentation. Keyboard shortcuts are provided for accessing the search, browse, and create interfaces for forms, files, and collections. Thorough (though now somewhat outdated) user-oriented

2.3. Features

óma *imitáá* *iiksikksimaatsáa'pii*
[om-wa](#) [imitáá](#) [iik-iksimatsi'ts-a'pii](#)
[DEM-PROX.SG](#) [dog](#) [INT-appreciative-be](#)
'that dog is happy to see you'

Figure 2.3: Screen shot of the Blackfoot OLD interface showing an IGT form display.

documentation—i.e., the *OLD User Guide*—is present in every OLD 0.2 application. Additionally, key parts of the interface have help buttons which bring up in-page instructions for accomplishing common tasks.

Figure 2.3 is a screen shot that shows a sentential form from the Blackfoot OLD displayed in interlinear glossed text format. The top line is an orthographic transcription. The bottom line is the sole translation. Lines two and three are the values of the morpheme break and morpheme gloss attributes. The colour-coded links indicate degree of consistency between morphological analysis and lexical items in the database. Blue links indicate perfect matches, green indicate partial matches, and the absence of a link indicates that there are no matches. That is, this representation shows that the database contains a lexical entry for the morpheme *imitáá* ‘dog’ (blue links mean perfect match) but none for *iksimatsi'ts* ‘appreciative’ (no links mean no match). The morpheme with the phonemic shape *a'pii* and the gloss ‘be’ is partially matched as indicated by the fact that ‘be’ is a green link and *a'pii* is not a link. This means that the database contains no forms with *a'pii* as their morpheme break value. However, it does contain at least

2.3. Features

one form with ‘be’ as its gloss value. If a user clicks on the green link, they will be presented with the nine lexical items glossed as ‘be’, one of which has the shape *a’p*. This suggests that the user may have mistranscribed the phonemic shape of this particular morpheme; or perhaps the lexical entries in the system are incomplete or inaccurate. Either way, the visual feedback provided by the GUI helps fieldworkers to see gaps and/or inconsistencies in their analyses. To further aid in the entry of lexically consistent morphological analyses, an in-page quick-search interface is provided. This performs searches asynchronously, i.e., without page refreshes and without locking the interface, so that lexical items, for example, can be looked up without leaving the form creation page. Thus, when creating the form in figure 2.3 the user could have used quick-search to look for either *a’pii* or *iksimatsi’ts* and found, prior to creation, that neither were present in the system.¹²¹

Though the OLD 0.2 GUI has many merits (such as those just discussed), it also reveals several opportunities for improvement and these have spurred the in-development GUI rewrite for the 1.0 version.

2.3.5.2 OLD 1.0 GUI

The OLD 1.0 GUI is currently under development and a prototype of the form add and browse interfaces has been implemented. The system is an independent client-side single-page application written in CoffeeScript, using the Backbone MV* framework, and jQuery tools for the UI widgets and

¹²¹The OLD 1.0 GUI will, during form creation, single out morphemes in the form that are not recognized and will prompt the user to have the system automatically create (all or a chosen subset of) them. This will allow contributors to focus on entering their phrasal and sentential data while automating the entry of the implicit lexical items and will greatly increase the rate at which these valuable lexical entries are created.

DOM manipulation. A more desktop-style experience is targeted, including a more consistent visual experience, expanded keyboard shortcuts (striving for pointer-less controllability), and conveniences such as auto-expanding input fields. Form browsing is much improved, with clickable buttons and keyboard shortcuts to scroll through, highlight, reveal hidden data of, and perform actions on form objects.

Numerous other improvements over the 0.2 interface are planned. Notable are tabular data view, and some level of offline capability and client-side data storage. Of course, a GUI for creating and using morphophonological models and parsers will need to be implemented. As will a GUI for corpora manipulation and treebank search, including tree representations of bracket-notation phrase structures. Also, certain (previously) server-side functionalities, e.g., orthography conversion and import/export, are being moved client-side. The technologies available for client-side browser-embedded applications have advanced rapidly in the past ten years and there are many exciting opportunities here.

2.3.5.3 OLD 1.0 API

The strength of the OLD 1.0 application programming interface (API) is its conceptual simplicity. Following the REST paradigm (cf. Fielding, 2000), OLD objects are viewed as *resources* that have a standard set of operations, and a corresponding set of patterns of HTTP methods and URIs for requesting them. The operations are search, create, read, update, and destroy (SCRUD). The method-URI pairs are *POST /object* for create, *GET /object* (and *GET /object/id*) for read, *PUT /object/id* for update, *DELETE*

2.3. Features

/object/id for delete, and *SEARCH /object* (also *POST /object/search*) for search. Data in and data out are, throughout, UTF8-encoded JSON-serialized associative arrays.¹²²

Aside from some idiosyncratic requests of particular objects, the description of the API in the above paragraph is sufficient to allow a developer to create programs that interact with any OLD application, effectively making an OLD application a sub-component of any program using it. This conceptual simplicity makes building tools on top of an OLD 1.0 web service an attractive proposition. And that, in and of itself, constitutes a potentially wide-ranging benefit to linguistic fieldwork.

2.3.6 Search

A core requirement of a successful linguistic fieldwork application is that relevant data be retrievable quickly and accurately. OLD applications fulfill this requirement by facilitating powerful searches over the data they contain. The OLD search functionality is, at its core, a simplified interface to the querying power of the underlying RDBMS, i.e., the relational database management system.¹²³ The searchability of the data set is enhanced by the OLD data structure, the auto-generation of values for particular attributes of forms, and the implementation of phrase structure search over treebank corpora.

¹²² An associative array is an abstract data type consisting of a set of key-value pairs. In Python these are called *dictionaries* and in JavaScript/JSON they are called *objects*.

¹²³Formulating searches based on conditions on relational attributes via SQL queries (i.e., performing joins) is prohibitively difficult for the majority of us who are not database wizards. The OLD simplifies relational search by hiding this particular complication from the user. While this results in some conceptual simplicity, it obviously comes at the cost of some expressivity in query formulation.

Just like a *where clause* in a structured query language (SQL) query, an OLD search is a hierarchy of filter expressions, i.e., it is a list of filters that are coordinated (conjoined or disjoined) and bracketed into a structure of unbounded complexity. A filter expression is a requirement that the value of a particular attribute possess a specified property, e.g., that the orthographic transcription begin with the character *t* or that the elicitor be among a specified set of users.

The OLD 1.0 search API expects search queries to be formatted as JSON arrays¹²⁴ (i.e., lists). The simplest such list is a 4-tuple where the first element is the name of an OLD object, the second the name of an attribute of that object, the third a relation (e.g., '='), and the fourth a pattern such that the relation holds between the pattern and the value of the object's attribute. These quaternary arrays are here dubbed *simplex filter expressions*. The JSON array in (4) (when sent in the body of a search request to the forms resource of an OLD web service) will return all forms with a(n orthographic) transcription attribute whose value is exactly *chien*. The OLD search array in (4) is equivalent to the SQL select query in (5).

(4) `["Form", "transcription", "=", "chien"]`

(5) `select * from form where transcription='chien';`

¹²⁴ OLD search arrays are here presented using the JavaScript Object Notation (JSON) syntax. An array is a sequence of comma-delimited elements enclosed in square brackets. Strings are enclosed in double quotation marks and numbers are represented without special formatting. Following standard conventions for Python class and attribute names, the names of OLD objects are written in camel case (e.g., a syntactic category object is referred to as `SyntacticCategory`) and the names of OLD object attributes are written in snake case (e.g., a syntactic category attribute is referred to as `syntactic_category`.)

2.3. Features

A more complex query is illustrated by the plausible Blackfoot search in (6). This retrieves all grammatical (actually, not ungrammatical) sentences containing a morpheme that is phonemically transcribed as *wa* or one that is glossed as one of five possible shorthands for *proximate*.¹²⁵ This is a pretty good approximation for a query that returns sentences containing the proximate suffix *-wa*.¹²⁶

```
(6) ["and", [{"Form", "syntactic_category", "name", "=", "S"},
            ["not", ["Form", "grammaticality", "=", "*"]],
            ["or", [{"Form", "morpheme_break", "regex",
                    "-wa( |-$)"],
                  ["Form", "morpheme_gloss", "regex",
                    "-(PROX|prox|Prox|PRX|prx)"]}]]]
```

Search (6) illustrates several important concepts relevant to OLD search construction: the syntax for constructing complex queries, conditions on relational attributes, and regular expressions.

The query in (6) is a complex filter expression constructed from simplex filter expressions that are composed with and coordinated via logical operators, i.e., conjunction "and", disjunction "or", or negation "not". As the query illustrates, a logical operator is always the first element of a binary array, which is itself a (complex) filter expression. Conjunction and disjunc-

¹²⁵In a collaboratively created database, it is sometimes necessary to create searches based on such disjunctive conditions. Of course, in an ideal situation, all contributors would gloss the same morpheme in the same way.

¹²⁶Of course, the *is not ungrammatical* condition might also be expressed using the inequality relation != instead of the negation operator. The example search is constructed this way in order to illustrate the syntax for negating filter expressions.

tion are followed by a list of filter expressions that constitute the conjuncts and disjuncts, respectively. Negation, on the other hand, is followed simply by the negated filter expression.

The first filter expression in (6) is a five-element (i.e., quinary) array that expresses a condition on a relational attribute, i.e., the syntactic category attribute of forms. The attribute is relational because its value is a reference to another object, viz., a syntactic category object. These objects have their own attributes, including the *name* attribute that is relevant to the condition currently under inspection. In contrast to the *[object, attribute, relation, pattern]* form of the quaternary OLD filter expression array, the quinary relational one is of the form *[object, attribute, attribute, relation, pattern]*. That is, the second element of the array is the name of the relational attribute (e.g., `syntactic_category`) and the third element is the name of an attribute of the object referenced by the relational attribute (e.g., `name`). Thus this filter expresses the condition that a form must have a syntactic category whose name is *S*.

The third and fourth simplex filter expressions of the query in (6) make use of regular expressions, hence *regex* as the name of the relations here. Regular expressions are powerful tools for succinctly expressing complex patterns over sequences. The string `-wa(|-|$)` is a regular expression pattern that matches *-wa* followed by a space, another morpheme delimiter (`-`), or the end of the string (`$`). The string `-(PROX|prox|Prox|PRX|prx)` matches the hyphen delimiter followed by any of five variously spelled shorthands for the word *proximate*. These regular expressions show how to use parentheses to group disjunctive options delimited by the vertical bar character `|` and

2.3. Features

how to match the end of a string using the dollar sign character \$. This just scratches the surface of what is made possible by regular expressions.

Of course, users of a graphical interface to an OLD web service should not be prompted to construct queries by composing JSON arrays as in (6). A sensible GUI will provide forced-choice fields where these are appropriate as well as visually intuitive interfaces for constructing the desired bracket/tree structures. Also highly useful will be in-page help with composing regular expressions. When completed, the GUI for the OLD 1.0 web service will allow users to define shorthands (i.e., macros) for oft-used regular expressions that can be used within search patterns. A practical example of this would be defining shorthands for regular expressions that match consonants, vowels, and syllables (relative to the inventory and phonotactics of the target language) and then using these macros to restrict searches to forms containing n -syllable words or even particular patterns within syllable i of a word.

The OLD objects that can be searched are forms, files, collections, corpora, sources, the set of forms remembered by a particular user,¹²⁷ languages, and searches of forms.¹²⁸ Since form objects are the most often searched, here follows a listing of the attributes that can be referenced by filter expressions over forms. For details on these attributes, see Appendix A. Searchable form attributes whose values are strings are tran-

¹²⁷Users can *remember* forms, meaning they can store references to specified forms in their OLD *memory*. The *memorizers* of a form are the set of users who have remembered the form, i.e., saved it to their memory within the system. The memory is simply a place where users can store a set of forms to be used for some other purpose, e.g., creating a text or for export.

¹²⁸Recall that searches over forms can be saved as objects in their own right. These form search objects can themselves be searched.

2.3. Features

scription, phonetic transcription, narrow phonetic transcription, morpheme break, morpheme gloss, comments, speaker comments, grammaticality, Universally Unique Identifier (UUID), syntactic category string, morpheme break identifiers, morpheme gloss identifiers,¹²⁹ break-gloss-category, syntax, and semantics. The value of the date elicited attribute is a date, and those of date-time entered and date-time modified are date-times. The value of the identifier attribute is an integer. Each of the following attributes has as its value another OLD object (the name of the object is indicated in parentheses, unless this is the same as the name of the attribute itself): elicitor (user), enterer (user), verifier (user), speaker, elicitation method, syntactic category, and source. Each of the following attributes has as its value a collection of other objects: translations, tags, files, collections, memorizers (users), and corpora.

The relations that can be specified in simplex filter expressions are equality (=), inequality (!=), regular expression (*regex* or *regexp*), like (*like*), is an element of (*in*), less than (<), greater than (>), less than or equal to (<=), and greater than or equal to (>=). The *like* relation matches strings against patterns that may contain two special wildcards, the underscore character `_` which matches any character one time and the percent sign character `%` which matches any character zero or more times. In practise, the *like* relation is often used for substring match, e.g., querying the database for forms with a translation transcription like `%hamburger%` to get all data that have *hamburger* in one of their translations.

¹²⁹The morpheme break identifiers and morpheme gloss identifiers values are system-generated JSON arrays storing references to other OLD forms that match the user-supplied morphemes and glosses in the morpheme break and morpheme gloss fields.

The *in* relation tests whether the value of the specified attribute is identical to any of the elements of the specified pattern array. For example, the single-filter *in*-based query in (7) returns all forms orthographically transcribed as *chien* or *chat*. The queries in (8) and (9) use the regular expression relation¹³⁰ and the disjunction operator, respectively, to return the same result set as that returned by (7).

(7) ["Form", "transcription", "in", ["chien", "chat"]]

(8) ["Form", "transcription", "regex", "^(chien|chat)\$"]

(9) ["or", [{"Form", "transcription", "=", "chien"},
["Form", "transcription", "=", "chat"]]]

The $<$, \leq , $>$, and \geq relations have obvious semantics when numbers are being compared; however, they also work with strings, dates, and date-time values. The query in (10) returns all forms with an identifier value less than 100, a date of elicitation before Christmas 2012, and an orthographic transcription that would be ordered before *abacus* in an alphabetic sort.

(10) ["and", [{"Form", "id", "<", 100},
["Form", "date_elicited", "<", "2012-12-25"],
["Form", "transcription", "<", "abacus"]]]

¹³⁰In regular expressions, the caret character \wedge matches the beginning of the string and the dollar sign $\$$ the end of the string.

2.3. Features

With only the search tools described so far, complex queries can be constructed and relevant forms (and other objects) can be accessed quickly.¹³¹ However, in addition to this the OLD enhances search based on morphological criteria via the system-generated values for the syntactic category string and break-gloss-category attributes of forms. Consider the hypothetical form (11) in an OLD application for French. Assuming that the database contains sensibly categorized lexical entries corresponding to those in its morphological analysis, (11)'s syntactic category string value will be (12).

(11) *Les chiens mangeaient*
le-s chien-s mange-aient
the-PL dog-PL eat-3PL.IMPF
'The dogs were eating.'

(12) D-Num N-Num V-Agr

The auto-valued syntactic category string attribute allows forms to be searched according to morphological patterns. For example a regular expression filter on syntactic category string values using the pattern in (13) will return forms consisting of a determiner (with zero or more suffixes), followed by a noun (with zero or more suffixes), followed by a verb (with zero or more suffixes). Thus, (11), or rather its syntactic category string value, will be matched by the regular expression in (13). A form such as *le chien mange* with a syntactic category string value of *D N V-Agr* will also

¹³¹OLD search requests can also specify the ordering of the results returned as well as a limit on the number of results returned (i.e., SQL *order by* and *limit*). The details of this are not discussed here.

be matched. This is a pretty good start for a search that returns sentences consisting of an intransitive verb with a simple DP subject.

(13) $\text{^D}(-[\text{^ }]+)? \text{N}(-[\text{^ }]+)? \text{V}(-[\text{^ }]+)?\text{\$}$

The key to understanding the regular expression in (13) is the component $(-[\text{^ }]+)?$ which matches zero or more suffix category names. The sub-expression $[\text{^ }]$ matches any character except a space. The plus sign character is a quantifier that means *one or more times*, so $-[\text{^ }]+$ means *match a hyphen followed by any character except a space, one or more times*. This is a sufficient characterization of one or more suffixes. Since we also want to allow for free morphemes without suffixes, we use the question mark character quantifier, meaning *zero or one time*, over the entire sub-expression (with scope indicated by the parentheses): $(-[\text{^ }]+)?$.

More fine-grained morphological patterns can be targeted by creating conditions on break-gloss-category values. This is because the phonemic shape, gloss, and categorial information that constitutes the morphological analysis of a form is all present here. Within the paradigm of concatenative morphology, a morphological analysis can be conceptualized as a list of words, each of which is a list of morphemes (interleaved by morpheme delimiters¹³²), each of which is a list of three elements, viz., shape, gloss, and category (14). The break-gloss-category value is simply a serialization of such a list of nested lists. The three components identifying a morpheme are

¹³²One could represent the morphological analysis of a word as simply a list of morpheme triples. However, since the choice of delimiter can have significance, this is, in general, an undesirable approach. One convention where choice of delimiter is significant is where the hyphen is used to indicate affixation and the equals sign for cliticization (cf. Bickel et al., 2008).

2.3. Features

delimited by an arbitrary (rare) delimiter—here represented as the vertical bar—and the serialized words are delimited by spaces (15).

(14) [[["le", "the", "D"], "-", ["s", "PL", "Num"]],
[[["chien", "dog", "N"], "-", ["s", "PL", "Num"]],
[[["mange", "eat", "V"], "-", ["aient", "3PL.IMPF", "Agr"]]]

(15) le|the|D-s|PL|Num chien|dog|N-s|PL|Num mange|eat|V-aient|3PL.IMPF|Agr

By performing regular expression searches on break-gloss-category values, conditions based on different aspects of the morphological analysis can be expressed simultaneously. For example, one could search for all forms containing a bi-morphemic word whose first morpheme is categorially *N* and whose second morpheme is glossed as *PL* (16).

(16) (|^)[^ -]+\|[^ -]+\|N-[^ -]+\|PL\|[^ -]+(|\$)

In (16), (|^) matches a space or the beginning of the string and, similarly, (|\$) matches a space or the end of the string. This ensures that we are matching a word. The sub-expression [^ -]+\|[^ -]+\|N matches a morpheme categorized as *N*, while [^ -]+\|PL\|[^ -]+ matches a morpheme glossed as *PL*. In these sub-expressions, [^ -]+ matches a string of one or more characters none of which are spaces or hyphens—this effectively matches a phonemic shape, a gloss, or a category name. The sub-expression \| matches the vertical bar, which must be escaped with a backslash so that the regular expression interpreter does not read it as signifying disjunction. The regular expression in (16) should return forms containing words an-

2.3. Features

alyzed as `chien|dog|N-s|PL|Num` as well as words such as *animaux* that might be analyzed as `anim|animal|N-o|PL|Num`.

The break-gloss-category values are especially valuable in that they allow for fine-grained targeting of specific morphemes by permitting simultaneous reference to a morpheme's shape, gloss, and category. This is useful for cases where two distinct morphemes have the same gloss (i.e., are synonymous) or the same shape (i.e., are homophonous) but the researcher wants to find forms containing only one of them. For example, depending on assumptions, Blackfoot may be analyzed as containing two homophonous *-wa* suffixes, one verbal and glossed as *3SG* and the other nominal and glossed as *PROX.SG*. By creating a search condition on break-gloss-category values using the regular expression in (17), one can retrieve with certainty only forms containing the verbal suffix.

(17) `["Form", "break_gloss_category", "regex", "-wa\\|3SG\\|Agr(|-|\\$)"]`

Note that attempting the same search by conjoining conditions on the morpheme break and morpheme gloss values, as in (18), will not suffice. This is because there is no guarantee that the phonemic shape match and the morpheme gloss match will correspond to the same morpheme. That is, the result set of the query in (17) will correctly exclude a form like (19) while the result set of (18) will incorrectly include (19).

(18) `["and", [{"Form", "morpheme_break", "regex", "-wa(|-|\\$)"}], [{"Form", "morpheme_gloss", "regex", "-3SG(|-|\\$)"}]]`

- (19) *Ipiima ki inihkiwa*
ipiim-a ki inihki-wa
enter-3SG and sing-3
'He came in and sang.'

A further improvement to search functionality is made possible by the corpus object introduced in the OLD 1.0. A corpus is, at its core, a list of forms. This list of forms can be written to a server-side file in accordance with a predefined set of *corpus generation* formats. At present, the only implemented corpus generation format writes to file each of the corpus' form's syntax values in their specified order and separated by newline characters. Assuming that these syntax values are phrase structure representations written in Penn Treebank-compatible bracket notation (cf. Marcus et al., 1993; Taylor et al., 2003) this generation format results in a treebank corpus that can be searched according to structural patterns using the TGrep2 utility (cf. Rohde, 2005). The effect is that OLD data sets can be searched according to complex syntactic criteria.

TGrep2 allows treebanks to be searched with reference to structural relationships between nodes, including whether one node dominates, immediately dominates, precedes, follows, or is a sister of another node. Regular expressions can also be used to match node names in a general manner. The example TGrep2 search expression in (20) returns all trees where a TP or an IP node (cf. the regular expression \wedge [TI]P\$) immediately dominates (<) a DP which itself dominates (<<) an AP. Thus an OLD form for *Le chien noir boit* 'The black dog is drinking' with syntax value (21) will be matched by the TGrep2 search pattern in (20). For details on what is possible with

TGrep2 structural searches, see Rohde (2005).

(20) /[^][TI]P\$/ < (DP << AP)

(21) (TP
 (DP (D le)
 (NP (NP (N chien))
 (AP (A noir))))
 (T'
 (T 0)
 (VP (V mange))))

Of course, structure-based search of OLD data sets is only possible if a significant subset of forms have valid syntax values. While some contributors may be motivated to create these manually, having the system generate them automatically is highly desirable. The OLD 1.0 already allows contributors to specify phonological and morphological models that can be used to create working generators and parsers. I hope to implement similar functionality with respect to syntactic modelling, with the ultimate goal of allowing for the creation of syntactic parsers that can help with the generation of structurally searchable syntactic representations. Analytic models and the parsers based on them, when integrated into a collaboratively created database like the OLD, constitute a unique opportunity for applying natural language processing techniques and formalisms to endangered language data sets for the benefit of fieldwork generally.

This section has shown how OLD applications facilitate the speedy and accurate retrieval of relevant data via a powerful search interface to a structured data set. Auto-generated supplementary morphological representa-

tions and structural search of treebanks further enhance the searchability of the data.

2.3.7 Automation

The OLD automates certain tasks in order to expedite the creation and retrieval of language data. The automatic generation of morphological representations via contributor-created parsers is discussed thoroughly in chapter 3. The automatic assignment of values to the syntactic category string and break-gloss-category attributes of forms is detailed in § 2.3.6. In comparison with the aforementioned, the following automations, though still useful, are minor: auto-insertion of previously used values during create and update, the automatic saving of state when deleting or updating certain objects, and the creation of reduced-size copies of large digital files.

The OLD 0.2 allows users to configure the system to automatically enter into the form create/update interface previously entered values for category, speaker, elicitor, verifier, source, and date elicited. This speeds up data entry. Also, as a contributor types words into a transcription field, the system attempts to guess values for the morpheme break and morpheme gloss fields based on previously entered morphological analyses. Once the graphical user interface for the OLD 1.0 is completed, users will be given the option to build and use particular morphological parsers to automate the creation of morpheme break and gloss values based on their transcriptions. However, to improve performance (i.e., to avoid time-costly parse requests to the server), the OLD 1.0 GUI will store and (at least as a first approximation) look up previously entered morphological analyses server-side.

Whenever a form or a collection is updated or deleted, all of its values prior to modification are saved to a backup table. This allows users to view previous versions of these data types and restore state, if desired. This is especially useful in a multi-contributor system where one user's data may be altered undesirably by another.¹³³ The OLD 0.2 currently allows users to view the history of a form, i.e., its previous states.

The OLD 1.0 can be configured to automatically create a lossy¹³⁴ copy of the digital file that constitutes the core of a file object. That is, large audio files in WAV format will be copied to the significantly smaller MP3 or OGG formats. Similarly, scaled-down copies will be created for large images. These smaller copies can be used when higher performance across the network is needed. The larger, lossless copies are still retained for when they are needed, i.e., for archiving and detailed (e.g., acoustic) analysis.

2.3.8 Consistency

The greater the consistency of a language database, the more effectively can relevant data be retrieved from it. The OLD seeks to promote consistency in the formatting of data while still allowing for variations that represent real differences in analysis. To this end, it provides features that encourage consistency of transcription, of morphological analysis, and of overall structure. The data structure of the OLD enforces a certain level of consistency in how

¹³³It may be desirable to implement a feature whereby a contributor can specify that certain or all of the forms that they enter not be modifiable by other users. At this point in time, the honour system has proved sufficient in this regard.

¹³⁴Lossy file formats are those which employ compression to reduce the size of the file yet result in information loss. File formats that do not result in data loss are termed *lossless*.

language data are carved up and labelled; the description of and argument for that structure is provided in § 2.3.4 and is not repeated here. Unicode normalization, input validation, orthography conversion, and morpho-lexical consistency feedback are features that are covered here.

2.3.8.1 Unicode normalization

The Unicode standard (Unicode Consortium, 2009) specifies a very large set of characters and assigns a unique code point, i.e., integer, to each one. There are a number of *encodings* that can be used to store strings of Unicode characters as binary data. OLD applications store string data using the UTF-8 encoding. Crippen (2010) is an excellent overview of Unicode with emphasis on its relevance to linguistics.

Unicode data can sometimes result in troublesome inconsistencies where equivalence classes are concerned. These are sets of characters and/or character sequences that are considered equivalent. For example, the Unicode standard defines single code points for both combining accent characters and for certain precomposed character-accent combinations. To illustrate, there is an equivalence between the precomposed character *á* (i.e., LATIN SMALL LETTER A WITH ACUTE, U+00E1¹³⁵) and the two-character sequence consisting of *a* (LATIN SMALL LETTER A, U+0061) followed by the COMBINING ACUTE ACCENT character U+0301. That is, *á* (U+00E1) is considered equivalent to *á* (U+0061, U+0301).

¹³⁵Note the convention of upper-casing names of characters. Also that of specifying a Unicode code point as a 4-digit hexadecimal number prefixed by *U+*. In fact, the standard defines more characters than $16^4 = 65536$ and it is not uncommon to see 5-digit hex code points.

2.3. Features

Issues arise because of inconsistencies in how different programs and operating systems handle these equivalence classes, with some opting for silent composition and others for decomposition. The result is that string data that may appear identical to the user are actually represented differently underlyingly and this can affect search. For example, searching a database for values containing *á* may (contrary to the searcher's desires) return results containing U+00E1 but not those containing U+0061, U+0301, or vice versa. An additional complication is that there is no precomposed counterpart to many sequences of base character followed by combining character, as is the case for acutely accented schwa *é* which can only be represented as the schwa character (U+0259) followed by the combining acute accent character.

The OLD handles this issue by normalizing *all* user input via the *Normalization Form Canonical Decomposition* algorithm better known by its acronym NFD. To return to the previous example, this means that an accented *a* character will always be stored within an OLD application as a sequence of base character *a* followed by the combining acute accent character, i.e., as U+0061, U+0301. Both input that results in the creation of data and input that constitutes search formulations are NFD-normalized in this way. This ensures that users will be able to search for data accurately without worrying that Unicode equivalence issues might be masking potential matches.¹³⁶

¹³⁶Note that Unicode NFD normalization has consequences for regular expression search. For example, the regular expression patterns `[ae]` and `(a|e)` are equivalent and will both match strings that contain either *a* or *e*. However, the regular expression patterns `[áe]` and `(á|e)` are not equivalent. The former will match *a*, the combining acute accent character U+0301, or *e*. The latter will match *á* or *e*, and is probably what is desired. Subtleties such

2.3.8.2 Input validation

An OLD application can be configured to restrict (or discourage) the use of certain characters and character sequences as values for certain form attributes, viz. the orthographic, narrow phonetic, phonetic, and morpho-phonemic transcriptions. This functionality thus enforces (or encourages) a certain degree of consistency in how linguistic forms are transcribed at all levels. In the OLD 0.2 graphical interface, the specified inventories of graphemes are used to generate clickable keyboard¹³⁷ widgets for entering all of the specified character sequences.¹³⁸

Administrators can specify inventories of graphemes for each of the four transcription-type attributes. A grapheme is a character or sequence of characters (i.e., a string). The system determines whether user input can be constructed by concatenating the graphemes (plus whitespace characters, punctuation, and delimiters, as appropriate). If it cannot, either the attempted change is disallowed and an error message returned or a warning message is issued.

Sometimes speakers will include words from other languages in their utterances and these words will be pronounced according to the phonology of the borrowed language. Often it makes sense for fieldworkers to transcribe these foreign words using the foreign language's orthography or using

as this need to be kept in mind when performing searches involving Unicode characters.

¹³⁷Such GUI “keyboards”, where non-standard characters are entered by clicking instead of typing, result in slow data entry and are therefore not a permanent solution. If a suitable OS-based keyboard layout is not available, then the Unicode keyboard layout editor Ukelele (created by SIL International) can be used to create one.

¹³⁸The OLD v. 0.2 also provides an interface which allows users to attempt to type the necessary graphemes and then provides feedback comparing the name and code point of the characters entered with those prescribed.

2.3. Features

phonetic characters not available in the validation inventories specified for the object language. Clearly this will be problematic if transcription input validation is set to restrict invalid input. The OLD handles this issue by allowing users to create forms tagged with the special *foreign word* tag. The system uses these foreign word forms to intelligently build exceptions into the input validation rules. For example, if *John* is a foreign word form in an OLD application for Blackfoot, then the system will allow a transcription like *Nitsínoaa anna John* ‘I saw John’ despite the fact that neither *j* nor *J* are present in the orthographic inventory specified for Blackfoot. Note that the presence of the *John* foreign word will *not* license a transcription like *Nitsínoaa anna Joan*; that will require the entry of a new foreign word form for *Joan*.¹³⁹

Transcription input validation is notably useful in cases where there are several similar characters that might be used for a single purpose. A case where this comes up in fieldwork on languages of the Pacific Northwest is where there are several similar diacritic combining characters that contributors might use in transcribing ejective consonants. For example, the COMBINING COMMA ABOVE RIGHT (U+0315) character and the COMBINING COMMA ABOVE (U+0313) character look very similar and both can be used to signify an ejective consonant. Input validation can help to ensure that ejectives are transcribed consistently.

¹³⁹If the foreign word participates in morphological or phonological processes of the object language, it should probably be transcribed in accordance with the object language. E.g., the fabricated Blackfoot pseudo-example *nitsigoogleatooma* ‘I Googled it’ should probably be transcribed as *nitsikooklatooma* or some such thing.

2.3.8.3 Orthography conversion

The OLD allows administrators to specify multiple grapheme inventories for the orthographic transcription attribute of forms and designate one as the system-wide storage orthography for such transcriptions. Users can then specify their own input/output orthography and use it to interact with the system wherever orthographic transcription values are concerned. The system transparently converts user input from the user-specific input orthography to the system-wide storage orthography and returns it in the user-specific output orthography. This facilitates consistency in orthographic representations while avoiding having to force contributors to adopt an unfamiliar orthography. As it is not uncommon for understudied languages to have multiple orthographies, orthography conversion can be a handy convenience.

The OLD 0.2 implements orthography conversion in the server-side logic. Orthographic transcriptions are first converted and then, if necessary, the converted transcription is validated as described in the above section. The OLD 1.0 web service still allows for the specification of multiple orthographies, but it leaves the actual conversion to client-side logic. That is, orthography conversion for the OLD 1.0 is not yet implemented and will be implemented in the in-development GUI for that system.

2.3.8.4 Morpho-lexical consistency feedback

OLD applications display the morphemes within the morpheme break and morpheme gloss values of forms as colour-coded links that indicate the degree to which the morphemes specified are already present as forms in the

2.3. Features

database. If there is a mono-morphemic (i.e., lexical) entry in the database that matches (both in terms of shape and gloss) a morpheme used in the analysis of a poly-morphemic form, then the embedded morpheme is displayed as a blue link to the matching lexical form. If the match is only partial, i.e., if only the shape or gloss matches, then the embedded morpheme is displayed as a green link. Hovering the mouse over the green link brings up a display of the partial matches, thus allowing the user to alter the analysis (if appropriate) without performing a separate search task.

Unicode normalization of input, input validation, orthography conversion, and visual feedback on the lexical consistency of morphological analyses all work together to promote a consistent data set. This consistency helps fieldworkers to quickly and accurately search for and otherwise retrieve relevant data from an OLD application.

2.3.9 Dictionary

The OLD 0.2 provides a very basic dictionary-like interface to the lexical items within an OLD application. The system assumes that a form is lexical if its morphological analysis lacks morpheme and word delimiters. These lexical forms are sorted and displayed according to the ordering of graphemes in the administrator-specified orthographic inventories. This convenience provides a familiar representation of (one characterization of) “the” lexicon¹⁴⁰

¹⁴⁰ *The* is in scare quotes because there may be several distinct lexica in an OLD application. For example, a single OLD application may be used by two distinct fieldworker groups. Group A may analyze certain verbal forms as morphologically simplex and may define an OLD-internal lexicon (using the OLD’s corpus construct) as containing those putatively simplex verbal forms. Group B, however, may assume morphological complexity within those same verb forms and may therefore choose to define lexica which do not include such putatively complex verbs but instead contain the morphemes (e.g., verb roots

that is implicit in an OLD application.

The OLD 0.2 dictionary implementation suffers, however, from the fact that it is, upon each new request, re-retrieved by means of a costly database query and its display representation regenerated. In order to improve responsiveness, the OLD web service or the GUI should be amended so that the representations of dictionaries are cached.¹⁴¹ Other issues with the OLD's support for dictionary creation are discussed above in the sections on Ling-Sync (§ 2.2.4) and FLEx (§ 2.2.3).

2.3.10 Documents

An OLD application allows users to create documents consisting of formatted prose and IGT-formatted representations of the form objects that are present in the database. Such documents are composed as *collection* objects. These objects have a number of attributes, but the most important among them is the contents attribute whose value is a string of text containing formatting commands in one of two lightweight markup languages (reStructuredText and Markdown) and references to form objects. The value of the contents attribute is converted to HTML when displayed and can be converted to XeLaTeX (whence PDF) when exported. The form objects referenced are displayed in IGT format in both the HTML and XeLaTeX/PDF outputs.

The forms referenced within the contents of a collection are also rela-

and transitivity affixes) within them.

¹⁴¹An alternative approach would be to create a separate web application dedicated to providing a dictionary interface to a lexicon defined as a corpus (or search) object of an OLD 1.0 web service. I leave this for potential future work.

2.3. Features

tionally associated to their collections, i.e., they constitute the value of a collection's *forms* attribute. In the OLD 1.0, one consequence of this is that searches over forms can restrict results to forms that are associated to one or more specified collections. That is, one can search *within* a collection. OLD collections can also have zero or more file objects associated with them via their *files* attribute. Finally, as of the OLD 1.0, collections can reference other collections within their *contents* values; this means that a series of small collections can be combined to create a large one.

Collections are useful tools for creating documents from the data present in an OLD application. Example use cases are representations of elicitation records, narratives, conversations, problem sets, lesson plans, chapters of grammars, and research papers. A collection representing a record of an elicitation session may, for example, contain the forms elicited during the session, notes by the researcher concerning the relevance of certain forms and ideas for future elicitations, and audio recordings of the entire session. Since collections can be exported as XeLaTeX¹⁴² (see below) they can easily be used as rough drafts for professional documents.

The OLD 1.0 includes a revised data structure for source objects which is effectively identical to the structure of BibTeX, the bibliographic database format commonly used with (Xe)LaTeX. This means that future versions of the OLD may allow users to cite sources within the contents of collection objects and so generate (XeLaTeX and HTML) documents that contain in-line citations and formatted bibliographies. This would further improve

¹⁴²For collections using reStructuredText markup, it is also possible to export into a format that is compatible with the Microsoft Word and OpenOffice.org word processors. Implementing this type of export is a planned feature.

2.3. Features

the usefulness of OLD collections in facilitating the speedy generation of professional (draft) documents using the data therein.

Currently, the OLD 0.2 allows users to export collections using a number of XeLaTeX-based formats. XeLaTeX is a typesetting program that allows users to write documents that contain commands which control how the document is to be structured and formatted. These XeLaTeX source files can be used to generate professionally formatted documents in PDF (and DVI) format. The benefit of using XeLaTeX as opposed to a what-you-see-is-what-you-get word processing program like Microsoft Word is that authors can focus on content and, by specifying a few parameters, let the typesetting program handle the formatting in a professional and consistent manner. XeLaTeX is very similar to the more familiar (and older) LaTeX, the primary difference being that XeLaTeX allows for Unicode characters in the source files. This is convenient for export from OLD applications since developers do not have to worry about writing general algorithms for translating Unicode characters to LaTeX commands.¹⁴³

The differences between the various XeLaTeX-based collection export formats has to do with which packages are used to format the interlinear glossed text representations of forms. The OLD 0.2 currently allows forms within collections to be IGT-formatted using either the Covington or ExPex packages. In addition to collection export, individual forms as well as search result sets of forms can be exported in a number of formats, includ-

¹⁴³Since XeLaTeX source files can be typeset as PDF documents via command line utilities, it would be relatively trivial to implement functionality that would non-mediate generate exportable PDF versions of collection objects. This is a planned feature for the OLD 1.0.

2.3. Features

ing XeLaTeX-based, several plain text-based, and a comma-separated values format that is importable into spreadsheet software like Microsoft Excel.

The OLD 1.0 does not currently implement any export functionality, either for collections or for (lists of) forms. Since all data returned by an OLD 1.0 web service are already structured as JSON objects, the task of generating export documents can easily be delegated to the user-facing applications, such as the browser-based GUI that is currently under development.¹⁴⁴

Functionality allowing contributors to *import* structured data into an OLD application is highly desirable but does not yet exist. Of course, since the OLD 1.0 is a web service that exposes a standards-compliant JSON/HTTP-based API, it would be relatively trivial to create command-line programs that take documents containing structured language data as input and use these to perform form create requests on live OLD applications. That said, the in-development GUI for the OLD 1.0 will include functionality that facilitates data import. This is especially useful for encouraging contributions from users who do not want to use an OLD application exclusively for all of their fieldwork tasks and would instead prefer an easy way to import the structured data produced by their preferred software. Such software might include FLEx, Shoebox, Toolbox, ELAN, LingSync, FileMaker Pro, OpenOffice.org database, RDBMSs (e.g., MySQL, SQLite, PostgreSQL, Oracle, etc.), spreadsheet programs, etc. The LingSync fieldwork application currently provides a graphical import interface that allows users to interactively map the structure of the file they are attempting to import to the

¹⁴⁴Typesetting XeLaTeX source files is not currently possible using browser-based (i.e., JavaScript-based) technologies. Such functionality would need to be implemented server-side within the OLD 1.0 web service.

application's data structure. Such functionality might be emulated by the OLD. Another option is to use comparisons of character sequence frequencies to intelligently guess the structural position appropriate to the components of the import data.¹⁴⁵

2.3.11 Access

Restrictions on access to data in a collaborative linguistic fieldwork application are important from two distinct vantage points. First and foremost are the requirements of the speakers and communities whose knowledge is encoded in the linguistic artifacts. Also important are the requirements of the fieldworkers who help to create the artifacts. This section describes and motivates the features of the OLD that enable administrators and contributors to restrict who has access to OLD language data and what kind of access is granted.

There are a range of reasons that speakers of endangered languages and their communities may have for wanting to restrict access to the data generated by fieldworkers and language consultants. It often happens that a speaker will speak quite candidly during a recorded elicitation session and may want to restrict access to all or parts of that recording for personal reasons. It also happens that particular stories or descriptions of rituals and cultural practices need to be restricted to just the language community or even to sub-groups within the community. Given the post-colonial context¹⁴⁶

¹⁴⁵This is an interesting approach which was suggested to me by fellow UBC Linguistics graduate student and fieldwork software developer Patrick Littell. He made effective use of it in a wiki-based application that was used in a UBC field methods course.

¹⁴⁶It is not unheard of that academic work related to indigenous culture and history can have unintended yet significant political and economic impact on the community

2.3. Features

of many fieldwork situations, there is an entirely understandable distrust of the dominant society that may result in speakers or communities adopting a generally cautious (or even zero tolerance) approach when it comes to sharing language data via a web-based application like the OLD. Whatever the particulars of the motivations, fieldworkers must communicate thoroughly and honestly about their intentions for sharing fieldwork data and come to an ethical agreement with speakers and communities concerning policies of data access.

Fieldworkers may also hold various positions on whether and the extent to which they want to share their fieldwork data. In my experience, the primary concern of fieldworkers in the context of a collaborative fieldwork application is that their data not be modified by other contributors without their knowledge.

The OLD provides a number of features which facilitate the creation of restrictions on access to data. The most basic is authentication. That is, only registered users with valid usernames and passwords can gain access to an OLD application. Passwords are encrypted using the Python module PassLib's implementation of Password-Based Key Derivation Function 2 (PBKDF2) using a salt and the default 20,000 iterations. The passwords are stored encrypted on the server, meaning that nobody (except somebody with the expertise to crack PBKDF2-encrypted passwords) can view the passwords in their unencrypted form. While I make no claim to cryptographic expertise, PBKDF2-based encryption is widely used in modern

concerned. I am thinking here of a case where a friend's master's thesis on a First Nations' traditional fishery management practices became a factor in a court decision concerning modern-day rights to harvesting that resource.

2.3. Features

technologies¹⁴⁷ and it can be safely assumed that the effort required to crack an OLD user's password would probably exceed the payoff for a would-be hacker. In short, breaking into an OLD application should be prohibitively difficult.

In addition to basic authentication, authorization to perform specified actions is determined by the value of the role attribute of the password-authenticated user. The roles defined by the OLD are administrator, contributor, and viewer. Administrators can do anything. Contributors can do everything except alter system-wide settings, create users, and modify restricted forms (see below). Viewers can only read data and can never create, update, or delete OLD objects. A form can be deleted only by the contributor who entered it or by an administrator. A form may be updated by any contributor or administrator. However, since the system creates a backup of a form whenever it is updated or deleted, previous state is never lost and can be restored.¹⁴⁸

In addition to the role-based authorization system there is a per-object authorization system. Objects tagged with the special *restricted* tag are considered restricted. These objects can only be read or modified by administrators and unrestricted users. Administrators define the set of unrestricted users (a subset of the contributors and viewers) as an attribute of the active

¹⁴⁷According to Wikipedia, PBKDF2 is used by Mac OS X Mountain Lion, Apple's iOS, Wi-Fi Protected Access (WPA and WPA2), and Microsoft Windows Data Protection API, among others.

¹⁴⁸The new GUI for the OLD 1.0 may provide a feature which would inform contributors whenever one of the forms they have entered or elicited has been changed, i.e., something similar to the activity feed of LingSync corpora. Such a feature could prove useful in field methods courses for allowing junior fieldworkers to be notified of their supervisors' comments on their work.

application settings object.

Taken together, the role-based and restriction-based authorization systems allow some level of nuanced control over who has what level of access to which data. The net effect is an extension in the hierarchy of user classification from administrators with the highest level on through unrestricted contributors, contributors, unrestricted viewers, and finally viewers, who have the lowest level of access.

There are some notable ways in which the authentication and authorization system of the OLD could be improved. Perhaps it should be possible for contributors to tag a subset of the objects that they enter as *private* and these would be viewable only to their enterers and to administrators. On the other end of the spectrum, it might be desirable to have the ability to tag certain objects as *unrestricted* and have these accessible even without password-based authentication.

The access restriction-facilitating features of the OLD can only contribute so much to practical solutions to this complex issue. Administrators and contributors need to communicate openly with their language consultants and the relevant communities and have clear agreements and protocols laid out for controlling access. As a result, it may be that certain data should simply not be entered into an OLD application. Other data may require restricted status. Having protocols in place for handling requests for access will save a lot of headaches.

In certain situations, and when appropriate conventions have been established, it may make sense for contributors to modify the information entered by others. For example, in order to keep analytical information ac-

curate and consistent, another user’s morpheme break, morpheme gloss, tag, or category information may be modified. If such modification would result in changes that the original elicitor does not agree with, then the form can be duplicated prior to modification. The modified duplicate will retain the enterer and elicitor values of the original and the modifier value will indicate which user has made the changes.¹⁴⁹

Note finally that the frameworks upon which the OLD are built—i.e., Pylons and SQLAlchemy—have built-in support for security against malicious attempts to access the system, e.g., via SQL injection attacks (Gardner, 2008).

2.3.12 Documentation

Good documentation is crucial. Good software that lacks it will, unfortunately but probably, languish. This dissertation is a high-level description of the OLD and an argument for its usefulness as a tool for linguistic fieldwork. However, it is not, in itself, sufficient as documentation. This section describes the current state of OLD documentation.

The OLD 0.2 applications that are currently being used have in-built user-oriented documentation in the form of an HTML document entitled the *OLD User Guide*.¹⁵⁰ This document provides an overview of the software, explaining the data structure and the interface for interacting with

¹⁴⁹The OLD 0.2 user interface has a feature which allows users to easily duplicate a form, a feature that will be re-implemented in the OLD 1.0 GUI.

¹⁵⁰The *OLD User Guide* can currently be found at <http://www.onlinelinguisticdatabase.org/help/olduserguide>. However, once the GUI for the OLD 1.0 has been completed, the [onlinelinguisticdatabase.org](http://www.onlinelinguisticdatabase.org) URL will resolve to a demo OLD 1.0 application that will contain a copy of the new documentation.

the system. While still useful for users of that version, the *OLD User Guide* is out-of-date even for the OLD 0.2.

Michael McAuliffe, a fellow UBC Linguistics graduate student, took the initiative to create a quick-start manual for the OLD 0.2 entitled the *OLD Quick Reference Guide*. This succinct guide for new users of OLD 0.2 applications can be found within certain language-specific OLD applications.

Detailed documentation for the OLD 1.0 has been written (Dunham, 2013a).¹⁵¹ This document, simply entitled *OLD Documentation: Release 1.0a1*, catalogues and explains the data structure, interface, and application logic of the OLD 1.0. It also provides detailed instructions on how to download and install the software. The code of the OLD is itself extensively documented using standard Python conventions. That is, modules, classes, methods, and functions all contain formatted documentation strings (a.k.a. *docstrings*). These docstrings are auto-assembled and incorporated into the *OLD Documentation* document as the final chapter entitled *API Documentation*. This chapter is useful as a reference for developers wishing to quickly understand implementation details of the system. Since all of this documentation is written in reStructuredText, it can be (and has been) converted to both HTML and PDF formats.¹⁵² The entire documentation (i.e., both the source reStructuredText documents and the generated HTML and PDF versions) comes packaged with the source which is available on GitHub and the Python Package Index.

¹⁵¹<https://online-linguistic-database.readthedocs.org>.

¹⁵²The HTML and PDF versions of the OLD 1.0 documentation can be accessed at <https://online-linguistic-database.readthedocs.org/> by clicking on the icon in the bottom right corner labelled “v: latest” and then clicking on “HTML” or “PDF”, respectively.

2.4 Summary

This chapter has described the OLD and argued that it is a useful and innovative tool for accomplishing linguistic fieldwork goals. The case for the OLD can be summarized as follows: it helps diverse groups of fieldworkers to build centralized and web-accessible stores of structured, consistent, exportable, highly searchable, and reusable data. In addition, it is well-documented and has been tested by dozens of fieldworkers against the idiosyncrasies of nine under-documented and/or endangered languages.

The next chapter (3) focuses on the OLD functionality which allows users to implement morphological and phonological models and to build automated analyzers (i.e., parsers) in order to expedite the creation of morphological representations and test analytical models against OLD data sets. This functionality can be viewed both as an additional point in favour of the OLD as a valuable tool for linguistic fieldwork as well as an example of how the insights, formalisms, tools, and methods of computational linguistics can contribute to linguistic fieldwork.

Chapter 3

Morphological Parser

Creator

This chapter describes the OLD *morphological parser creator*, an application added to the core OLD application (as described in chapter 2), which allows users to create morphological parsers using OLD data. The morphological parser creator takes rewrite rules and OLD corpora as input and creates a morphological parser as output. The parser creator uses finite-state transducers (FSTs) to model the morphophonology and N -gram language models (LMs) to rank candidate parses.

The incorporation of morphological parsers into the OLD contributes to the software's central goal of facilitating fieldwork by i) expediting morphological analysis creation and ii) automating the testing of linguistic analyses against data sets that, for understudied languages, are relatively large. Chapter 5 presents the results of using the OLD morphological parser creator to build parsers for the Blackfoot language.

This chapter is structured as follows. § 3.1 provides a high-level overview of the architecture of the morphological parsers created by the application. § 3.2 provides the background on finite-state machines and formal language

theory for the subsequent sections which explain how to use the application to build phonologies (§ 3.3), morphologies (§ 3.4), and morphophonologies (§ 3.5). Finally, § 3.6 reviews N -gram LMs and explains how to use the OLD morphological parser creator to build parse candidate rankers on top of these.

3.1 Architecture

The term *morphological parser* here refers to a computer program that, when given a transcription (phonetic or orthographic) of a word, returns a representation of its morphological analysis. This section provides a high-level overview of the components of parsers created by the OLD morphological parser creator.

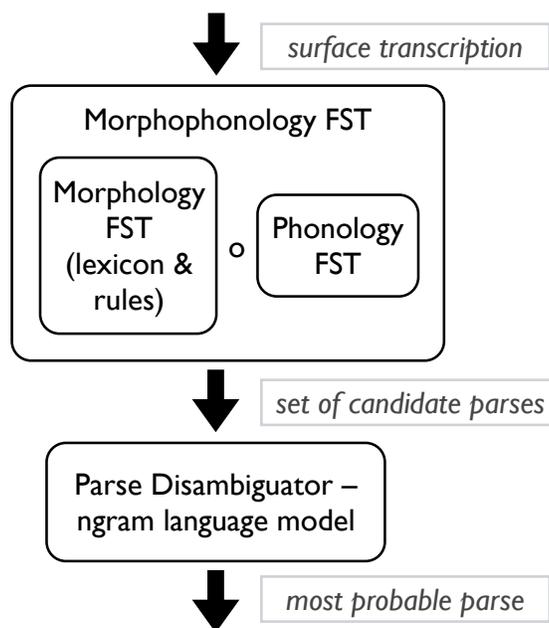


Figure 3.1: Schema of an OLD morphological parser.

Figure 3.1 schematizes the architecture of an OLD parser. The two highest-level components are the FST that implements the morphophonology and the N -gram LM that serves as the parse disambiguator.¹⁵³

The morphophonology takes a transcription of a word as input and returns a set of sequences of morphemes and delimiters, i.e., candidate morphological analyses. A morphophonology is the composition of a phonology FST and a morphology FST. A phonology analyzes a surface (i.e., phonetic

¹⁵³I use *ranker* and *disambiguator* interchangeably for this component. The N -gram LM ranks a candidate set by assigning a probability to each parse candidate. By selecting the most probable parse we disambiguate the candidate set.

or orthographic) representation and returns a set of sequences of morphemes. A morphology filters out those morpheme sequences that are incompatible with a given lexicon and a given set of morphotactic rules. The parse disambiguator returns the most probable parse from the set of candidates returned by the morphophonology.

An FST is a formal model that encodes a regular relation, i.e., a set of ordered pairs of strings where the first element of each pair is a member of a given regular language and the second element is a member of another regular language. Phonological transformations and morphotactically valid sequences of morphemes can both be expressed as regular relations and, as a result, as FSTs (Johnson, 1972). In addition, any sequence of FSTs can be composed to form a single FST that performs the same mapping as serial application based on the original sequence (Schützenberger, 1961). Since FST-based analysis and generation are computationally tractable transformations on strings, and since there are efficient algorithms and implementations for compiling CS phonological rewrite rules to FSTs (Karttunen et al., 1992), it is easy for linguists with some minimal training in rule-based phonology to specify phonological and morphological components that can be used to create computer programs that efficiently parse surface transcriptions to sequences of morphemes.

N -gram LMs—i.e., the tools used to disambiguate the outputs of morphophonological analysis (i.e., select a best candidate from multiple)—are data structures that can be used to assign probabilities to sequences of morphemes, i.e., morphologically analyzed words (cf. Manning and Schütze, 1999).

3.1. Architecture

By creating interfaces to the rule-to-FST compiler *foma* (Hulden, 2012) and the N -gram LM estimator *MITLM* (MITLM, 2013), and by integrating fieldwork data sets into these components, the OLD morphological parser creator facilitates the task of constructing fully functional morphological parsers. This involves, minimally, writing a phonology script as a sequence of CS rewrite rules and specifying three corpora: one for extracting a lexicon, another for extracting morphotactic rules, and a final one for calculating N -gram probabilities.

This approach to creating morphological parsers is justified within the context of the OLD because it leaves the fieldworker in control of the phonology and the morphology. Since fieldworkers tend to be experts (or at least sufficiently knowledgeable) in these domains, such control allows for the creation of parsers that are both practical fieldwork tools—insofar as they expedite the creation of morphological analyses—and effective research tools—insofar as they allow for specific phonological and morphological models to be implemented computationally and quickly tested against rare data sets.

The OLD morphological parser creator illustrates one particular type of approach to creating morphological parsers. While it does, as just mentioned, have the benefit of giving linguists fine-grained control over the underlying models, other approaches (e.g., Gildea and Jurafsky (1996); Goldsmith (2001)) could result in more robust and better performing parsers. Some deficiencies of the current approach include the fact that the morphology component (§ 3.4) cannot handle unseen category sequences or unseen morphemes.¹⁵⁴ The parser creator application described here should be viewed

¹⁵⁴It should also be made clear that since the morphological parser creator is part of

as a module which has been added to the core functionality of the OLD. Additional morphological parser creation modules may, in the future, be developed and added to the OLD in order to complement and/or improve on the functionality described here.¹⁵⁵

3.2 Finite-state machines & grammars

This section provides an overview of finite-state machines (i.e., automata and transducers), formal grammars, and the languages and relations that they describe. It may prove a useful reference for the sections on OLD phonologies (§ 3.3), morphologies (§ 3.4), and morphophonologies (§ 3.5) that follow.

A regular grammar is a set of rules for generating a regular language, i.e., a set of strings. Formally, a regular grammar is a four-tuple (N, Σ, P, S) , where N is a finite set of non-terminal symbols, Σ a finite set of terminal symbols (including the empty string ϵ), P a finite set of production rules, and $S \in N$ the unique start symbol. The set of production rules of a regular grammar are of the form $N \rightarrow \Sigma^* N$, i.e., rules where a non-terminal symbol expands to a string of zero or more terminal symbols followed by a non-

the OLD version 1.0 and since that version currently lacks a production-ready GUI, *there is no GUI for the parser creator or for the parsers created by means of it*. This means that parsers created by this application cannot currently be used to suggest parses (or ordered lists of candidate parses) to users of an OLD application during data entry. Such interface-integrated parse suggestions are a crucial ingredient in making these parsers useful to fieldworkers and this functionality is a major planned feature for the GUI for the OLD 1.0 that is currently under development.

¹⁵⁵For the sake of brevity, I sometimes refer to components of the parsers created using the OLD morphological parser creator as simply OLD phonologies, OLD morphologies, etc. These should be understood as shorthands, e.g., for *phonology created using the OLD morphological parser creator*, etc. and should not be taken to imply the preclusion of other types of parsers or parser components within the OLD.

terminal (cf. Roche and Schabes, 1997).¹⁵⁶

The regular grammar in (22) generates the (regular) language consisting of the set of strings that begin and end with a and have zero or more b s in the middle, as represented in (23).

$$(22) \quad S \rightarrow aBa$$

$$B \rightarrow bB$$

$$B \rightarrow \epsilon$$

$$(23) \quad \{aa, aba, abba, abbba, abbbba, \dots\}$$

The regular language generated by the regular grammar (22) can also be expressed by the regular expression¹⁵⁷ in (24).

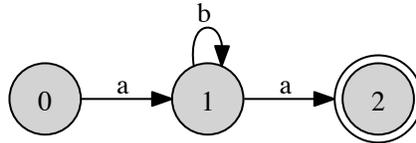
$$(24) \quad a b^* a$$

All and only the strings generated by (22, 24) will be recognized by the FSA depicted in the network diagram in Figure 3.2. In these network diagrams, states are represented by circles and transitions by directed arcs. State 0 is the unique start state and all states with double borders are final states.¹⁵⁸

¹⁵⁶Technically, the restriction that production rules be of the form $N \rightarrow \Sigma^*N$ results in a *right* regular grammar. Grammars where all production rules are of the form $N \rightarrow N\Sigma^*$ are *left* regular grammars.

¹⁵⁷The regular expression syntax in (24) is that accepted by foma (and XFST). It is similar but not identical to that accepted by the regular expression parsers included in many programming languages. The primary difference has to do with the treatment of whitespace. In the foma syntax, a sequence of adjacent characters are parsed as a single symbol while in other regular expression parsers this is not the case. As a result, the foma regular expression $a b^* a$ is equivalent to the Python regular expression ab^*a while foma ab^*a is equivalent to Python $(ab)^*a$.

¹⁵⁸These network diagrams are created using foma.

Figure 3.2: FSA network diagram for $a b^* a$.

An FSA can be thought of as a recognizer or characteristic function on strings. In order to use an FSA network diagram to recognize a string, begin in q_0 (i.e., the initial state) and at the left edge of the string to be recognized. Move from state to state by following arcs and consuming from the string one of the symbols labelled on the arc. Continue this process of transitioning between states until the string has been completely consumed or no further transitions are possible. If the string has been consumed and a final state has been reached, then the FSA recognizes the string. In order to generate strings, begin at q_0 with an initially empty string. Then follow an arbitrarily chosen arc and suffix one of its labelled symbols to the string. Continue this process an arbitrary number of times and stop on a final state.

Formally, a (deterministic¹⁵⁹) FSA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite set of symbols, δ is the transition function (from ordered state-symbol pairs to states, i.e., $Q \times \Sigma \rightarrow Q$), q_0 is the start state, and F is the set of accept states (cf. Hulden, 2009; Beesley

¹⁵⁹A deterministic FSA is one wherein “no state has more than one outgoing arc with the same label” (Beesley and Karttunen, 2003, p. 75).

and Karttunen, 2003). The FSA represented by Figure 3.2 is given in (25).

$$(25) \quad (\{q_0, q_1, q_2\}, \{a, b\}, \\ \{(q_0, a) \rightarrow q_1, (q_1, a) \rightarrow q_2, (q_1, b) \rightarrow q_1\}, q_0, \{q_2\})$$

Now consider the regular relation illustrated in (26), i.e., the infinite set of ordered pairs where the first element is a string from the universal language¹⁶⁰ and the second is that same string with every *a* replaced by a *b*, except when the *a* is preceded by a *b*.

$$(26) \quad \{(pineapple, pinebpple), (banana, banbnb), (plum, plum), \dots\}$$

If R is the regular relation expressed by an FST, then let dom be the domain of R , i.e., $dom = \{x | \exists y : (x, y) \in R\}$, and let ran be its range, i.e., $ran = \{y | \exists x : (x, y) \in R\}$. *Generation* (a.k.a. *downward application*) is the process of providing a string $s \in dom$ as input and receiving as output $\{y | (s, y) \in R\}$. The converse operation is *analysis* (a.k.a. *upward application*) which involves providing as input $s \in ran$ and receiving as output $\{x | (x, s) \in R\}$ (cf. Beesley and Karttunen, 2003).¹⁶¹

Using the foma syntax,¹⁶² the regular relation in (26) can be expressed by the rewrite rule in (27).

¹⁶⁰The universal language is the set of all finite strings and is expressible in foma regular expression syntax as `?*` and in conventional regular expression syntax as `.*`.

¹⁶¹In general when working with FST jargon, it is useful to simply memorize the association of *left*, *upper*, and *underlying* on the one hand and *right*, *lower*, and *surface* on the other. That is, the left-hand side of an ordered pair (a, b) is the upper side and analysis (a.k.a. parsing, upward application, or lookup) is the process of mapping a string from the right-hand/lower side to one from the left-hand/upper side.

¹⁶²What is here termed the *foma syntax* was actually first developed for the Xerox Finite-State Transducer program, i.e., `xfst`. Kaplan and Kay (1994) provides an algorithm for compiling CS phonological rewrite rules to FSTs.

$$(27) \quad a \rightarrow b \mid \backslash b _$$

The notation of (27) is very similar to the CS rewrite rules of Chomsky and Halle (1968) and subsequent work in rule-based phonology. Such rules are of the form $\alpha \rightarrow \beta \mid \delta _ \gamma$, i.e., α is rewritten as β whenever it follows δ and precedes γ , where α , β , δ , and γ are arbitrarily complex strings that may be empty. In the foma rewrite rule syntax, the functional symbols are slightly different: \rightarrow is \rightarrow , \mid is $\mid\mid$, and $_$ is $_$. In addition, α , β , δ , and γ are regular expressions in the foma syntax. For example, the left-hand side context of (27) is the regular expression $\backslash b$ which denotes the “term complement language” (Beesley and Karttunen, 2003, ch. 2) of b , i.e., the set of all single-character strings minus b .

That the transformations expressed by phonological CS rewrite rules can also be expressed as regular relations is an important finding. Since any regular relation can be expressed as an FST and since an FST can be used to efficiently map strings from the relation’s domain to its range (i.e., generation) or vice versa (i.e., analysis), the finding means that computationally tractable parsers and generators can, in principle, be created on the basis of CS phonological rewrite rules. This is a surprising result since unrestricted CS grammars are more expressive (i.e., can be used to generate a wider range of languages) than the regular grammars that generate the domains and ranges of regular relations. However, Johnson (1972) showed that phonological rewrite rules always tacitly assume that the site of application moves left or right after each application and that this results in the equivalence with regular relations. For example, the rule $i \rightarrow ii \mid s _$ will

generate sii from si but it will not generate $siii$ or $siii$, etc. from that same input (cf. Beesley and Karttunen, 2003, ch. 5).

An FST that implements the regular relation (26) is depicted in the network diagram in Figure 3.3. The labels on the arcs are ordered pairs of strings: \mathbf{a} and \mathbf{b} are shorthand for (a, a) and (b, b) , respectively; $\langle \mathbf{a:b} \rangle$ is a notational variant of (a, b) ; and $\textcircled{\@}$ is shorthand for $\{(e, e) : e \notin \{a, b\}\}$.

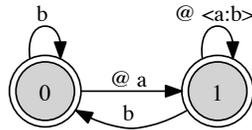


Figure 3.3: Network diagram for $\mathbf{a} \rightarrow \mathbf{b} \mid \mid \backslash \mathbf{b} _.$

Thus, to perform downward application (i.e., generation) on bana using Figure 3.3, begin in q_0 , read b , write b , and return to q_0 . Then read a , write a , and move to q_1 . Then read n , write n , and stay in q_1 . Then read a , write b , and stay in q_1 . The result is banb .

In the interfaces of *foma* and *XFST*, the command `apply down` (or just `down`) is used to generate and the command `apply up` (or just `up`) is used to analyze (or parse). Using the example FST diagrammed in Figure 3.3, `apply down banana` will yield $\{\mathit{banbnb}\}$ while `apply up banbnb` will yield $\{\mathit{banbnb}, \mathit{banbna}, \mathit{bananb}, \mathit{banana}\}$. The *apply up* and *apply down* terminology is used in the interfaces to the parser-related resources that are exposed by the OLD.

The formal definition of an FST is a 5-tuple $(Q, \Sigma, q_0, \delta, F)$, where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, δ is a partial mapping from $Q \times (\Sigma \cup \{\epsilon\})$ to $Q \times (\Sigma \cup \{\epsilon\})$, and $F \subseteq Q$ is the set of final states (cf. Hulden, 2009; Beesley and Karttunen, 2003). The FST depicted in Figure 3.3 is defined in (28).¹⁶³

$$(28) \quad (\{q_0, q_1\}, \{a, b\}, q_0, \\ \{(q_0, b) \rightarrow (q_0, b), (q_0, a) \rightarrow (q_1, a), (q_0, @) \rightarrow (q_1, @), \\ (q_1, @) \rightarrow (q_1, @), (q_1, a) \rightarrow (q_1, b), (q_1, b) \rightarrow (q_0, b)\}, \{q_0, q_1\})$$

More sophisticated and linguistically relevant foma rewrite rules (and their equivalent FSTs) are discussed in the sections that follow and, in particular, in the exposition of a phonology FST for Blackfoot given in § 5.2.1. For further reference on FSTs, their specification via CS rewrite rules as in (27), and practical examples related to natural language morphology and phonology, see Beesley and Karttunen (2003). While that text assumes the proprietary X(erox) FST software, the interface exposed by foma is nearly identical and most of the examples can be tested and explored using this open source and freely available alternative.

3.3 Phonologies

An OLD phonology is a mapping between underlying and surface representations. The practical purpose of a phonology within the context of a parser built using the OLD morphological parser creator is to analyze phonetic or

¹⁶³Here @ represents any symbol not present in Σ , i.e., $@ = \{x | x \notin \Sigma\}$. In mappings, it represents the same symbol on both sides of the arrow. That is, $@ = \{(x, x) | x \notin \Sigma\}$.

3.3. Phonologies

orthographic transcription values to morpheme break values.¹⁶⁴ That is, a phonology may encode either canonical phonological transformations (i.e., from phonemes to phones) or spelling rules.

An OLD phonology is an ordered set of rewrite rules that is specified via the foma regular expression/rewrite rule syntax introduced in § 3.2 (cf. Hulden, 2012; Beesley and Karttunen, 2003).¹⁶⁵ This syntax accepts statements of the form provided in (29), i.e., the `define` keyword, followed by the name of the FST to be defined, followed by the FST definition in CS rewrite rule format, and terminated by a semicolon.

```
(29) define <name> <FST-definition> ;
```

The only requirement imposed by the OLD is that the phonology script define an FST named `phonology`. Typically, the definition of the `phonology` FST will be the composition of an ordered list of previously defined FSTs representing phonological rules.

Consider *breaking*, a Blackfoot phonological rule defined in Frantz (1991) and repeated in (30). Since the /kI/ sequence never, to my knowledge, occurs morpheme-internally,¹⁶⁶ we can rewrite this rule as a transformation on the morpheme delimiter “-”, as in (31).

¹⁶⁴Actually, as discussed below, the underlying representation that is the input to phonological generation or the output of phonological analysis can also be a string representation of sequences of richly represented morphemes. This means that certain OLD phonologies can be viewed as analyzing transcription values to *break-gloss-category* values.

¹⁶⁵A reference document for foma regular expressions is available at <https://code.google.com/p/foma/wiki/RegularExpressionReference>.

¹⁶⁶Frantz (1971, 1978) and, to a lesser extent, Frantz (1991) use the capital “I” to denote a phoneme with identical features to /i/, i.e., a high front unrounded vowel, but which causes breaking in the immediately preceding velar stop. This phoneme is not used in (Frantz and Russell, 1995); instead, example sentences are provided to indicate “breaking” /i/ for those entries where this is necessary.

(30) $k \rightarrow ks \mid _ I$

(31) $_ \rightarrow s \mid k _ I$

Using the foma rewrite rule syntax, the statement in (32) defines an FST that implements the breaking rule as formulated in (31).

(32) `define breaking "-" -> s | k _ I ;`

Aside from some superficial notational differences, the foma statement in (32) is essentially no different from the rules expressed in (30, 31). Anyone familiar with type of CS rewrite rules popularized in *The Sound Pattern of English* (SPE) (Chomsky and Halle, 1968) for the expression of phonological transformations should have little problem adjusting to the foma conventions (cf. Hulden, 2012; Beesley and Karttunen, 2003).¹⁶⁷

A network diagram for the FST that implements the breaking rule in (32) is provided in Figure 3.4.

¹⁶⁷The hyphen-minus symbol -, which is used to indicate morpheme boundaries in morphophonemic transcriptions, must be enclosed in quotation marks since it is a reserved symbol. For the complete list of foma reserved symbols, see https://code.google.com/p/foma/wiki/RegularExpressionReference#Reserved_symbols.

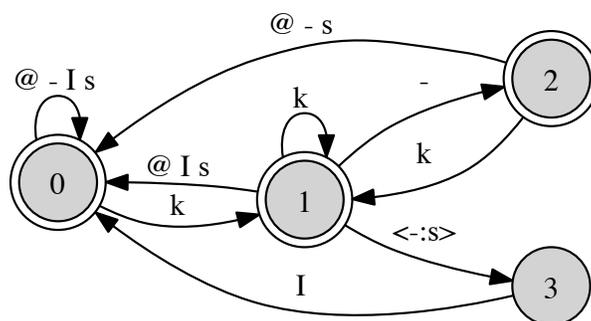


Figure 3.4: Finite-state transducer (FST) network diagram for "-> s | | k _ I ;".

The foma script defined in Figure 3.5 is a valid OLD phonology since it defines a **phonology** FST. In total, it defines three FSTs: **breaking**, which transforms /k-I/ into [ksI], **delimiterDelete**, which deletes the morpheme delimiter /-/, and **phonology**, which defines the phonology FST as the composition (.o.) of **breaking** and **delimiterDelete**. Such a phonology will generate [iiksIoyii] from /iik-Ioyii/ and will analyze (parse) [iiksIoyii] to an infinite¹⁶⁸ set of strings that contains /iik-Ioyii/ as a member.

¹⁶⁸The **delimiterDelete** FST is responsible for the infinite parse set. In generation, this rule maps a hyphen-minus to the empty string (here represented by 0) in any context; in parsing, it maps any empty string to a hyphen-minus in any context. Thus **delimiterDelete** will generate [iikihpiyi] from /iik----ihpiyi/ and will parse [iikihpiyi] into an infinite set of strings that contains /iik----ihpiyi/ as a member.

3.3. Phonologies

```
define breaking "-" -> s || k _ I;  
define delimiterDelete "-" -> 0;  
define phonology breaking .o. delimiterDelete;
```

Figure 3.5: Partial Blackfoot phonology.

The OLD interface to phonology objects allows multiple phonologies to be created, retrieved, updated and deleted. Thus users can create phonologies that are attuned to particular analytic assumptions, speakers, dialects, etc.

In addition to the Create, Read, Update, and Delete (CRUD) operations that are standard for all OLD resources, OLD phonologies expose the following: compile, serve compiled, apply down, and run tests. Issuing a request for an OLD phonology to be compiled results in the spawning of a new Python thread which, in turn, creates a foma process to compile the phonology script, i.e., convert it to a binary format that can be used to quickly convert strings to and from underlying and surface representations. The request results in an immediate response containing the JSON representation of the phonology. The requester must then periodically request (i.e., poll) the phonology's data until the value of its `compile_attempt` attribute¹⁶⁹ has changed, indicating that the attempt to compile has terminated.¹⁷⁰ If the value of the `compile_succeeded` attribute is `True`, then the phonology

¹⁶⁹This value is a UUID that is changed upon termination of each compile attempt.

¹⁷⁰Depending on the complexity of the FST defined, the duration of the one-time *compilation* event can be considerable, i.e., hours in some cases. However, FST *application* (i.e., parsing or generation) time using a precompiled FST should take a fraction of a second.

was successfully compiled.

Once a phonology has been compiled, a *serve compiled* request will serve the compiled binary file for download and local use. An *apply down* request on a compiled phonology will respond with a JSON object containing the surface representations corresponding to the underlying representations supplied in the request.

As a convenience, the OLD recognizes foma script comments of a certain form as test declarations. When *run tests* is requested against a particular phonology, the OLD will extract the test declarations from the script, run the tests, and respond with the results. This allows a user to define a data set that their phonology must account for and then tweak the script until all tests pass.¹⁷¹

A test declaration comment is a line that begins with the string `#test`. This string should be followed by two words, the first being an underlying representation and the second a surface representation that the phonology should generate from it. For example, the simple phonology in Figure 3.5 can be extended to include two tests by adding two new comments as in Figure 3.6. The OLD parser creator will determine whether the phonology generates `iikihpiyi` from `iik-----ihpiyi` (it does) and `iiksoyii` from `iik-Ioyii` (it does not) and respond with a summary of test results. Note that in order for a test to pass it is only necessary that the phonology generate the anticipated surface form for the given underlying form. That is, a phonology may pass a test even if it generates additional unanticipated

¹⁷¹The system *should* also allow users to automatically test a phonology against a corpus of forms. That is, this is a feature that should be, but has not yet been, implemented.

3.4. Morphologies

surface forms or if it parses a given surface form to underlying forms that are unanticipated.

```
#test iik-----ihpiyi iikihpiyi
#test iik-Ioyii iiksoyii
define breaking "-" -> s || k _ I;
define delimiterDelete "-" -> 0;
define phonology breaking .o. delimiterDelete;
```

Figure 3.6: Partial Blackfoot phonology with test declarations.

3.4 Morphologies

Like an OLD phonology, an OLD morphology is, at its core, a foma script that defines an FST. While a phonology analyzes surface transcriptions to underlying ones, a morphology's primary purpose is to recognize valid outputs of phonological analysis, i.e., licit sequences of morphemes and delimiters.

OLD morphology resources also have an extended interface. In addition to the standard CRUD requests, the following are also defined: generate, generate and compile, serve compiled, apply down, and apply up. Aside from those involving generation, which are discussed below, the effect of all of these requests should be clear from the discussion of the phonology interface (§ 3.3).

It is possible to manually compose a morphology FST script. However, since anything beyond a toy morphology will require the listing of thousands

3.4. Morphologies

of lexical items, manual creation is generally impractical. Therefore, the interface for creating an OLD morphology requires specification of a corpus from which lexical data can be extracted and another (perhaps the same) from which morphotactic rules (and, optionally, also lexical items) can be extracted.¹⁷² Requesting that a morphology be *generated* means asking the OLD to create a foma script on the basis of the lexicon and rules corpora referenced by the morphology.

Once lexicon and rules corpora have been specified for a morphology, it is necessary to decide whether the outputs of generation and/or analysis are to be rich or impoverished. A rich representation includes shape, gloss, and category information of morphemes while an impoverished one includes only shape information. Let *upper* refer to the outputs of analysis, i.e., the strings in the domain of the regular relation expressed by the morphology FST. Let *lower* denote the outputs of generation, i.e., the range. Using this terminology, we have the four options schematized in Table 3.1.

The values in the *analysis example* column of Table 3.1 illustrate the effect of issuing an **apply up** command on a specified input. For example, with a *rich upper, impoverished lower* morphology (*ruil*), an impoverished shape-only representation of a morphological analysis, e.g., *chien*, will be parsed/analyzed to a rich representation, e.g., *chien|dog|N*.¹⁷³

¹⁷²The OLD *does* allow for morphology creation wherein the morphotactic rules are manually specified (i.e., instead of being extracted from a rules corpus). Under this option, the rules are to be provided as a string where whitespace-delimited words are specified as sequences of morpheme category names and morpheme delimiters, e.g., **D D-PHI N N-PHI V V-AGR V-TNS-AGR**. While morphotactic rules can be specified manually, lexical classes currently cannot. Therefore, it is not at present possible to manually specify an OLD morphology in its entirety.

¹⁷³A rich representation of a morpheme is an ordered triple (m, g, c) which is represented as a string by joining the three elements with a rare delimiter. In the present exposition,

3.4. Morphologies

abbr.	< upper :	lower >	analysis example
<i>rurl</i>	rich	rich	<i>chien dog N</i> → <i>chien dog N</i>
<i>ruil</i>	rich	impoverished	<i>chien</i> → <i>chien dog N</i>
<i>iuil</i>	impoverished	impoverished	<i>chien</i> → <i>chien</i>
<i>iurl</i>	impoverished	rich	<i>chien dog N</i> → <i>chien</i>

Table 3.1: Morphology types.

The two options with identical values for upper and lower (i.e., *rurl* and *iuil*) result in morphologies which are simply recognizers. Options *ruil* and *iurl*, on the other hand, not only recognize valid sequences of morphemes, they transform their inputs during analysis (and generation), adding information in the case of *ruil* and removing information in the case of *iurl*.

The options where the inputs to morphological analysis are impoverished (*ruil* and *iuil*) are compatible with standard phonologies, i.e., those which analyze/parse surface representations into sequences of morpheme shapes and delimiters. The options where the inputs to morphological analysis are rich (*rurl* and *iurl*) require phonologies that parse surface representations into rich representations of morphemes and delimiters. Such phonologies are more complex and opaque yet are more expressive insofar as they allow for the expression of transformations that take categorial and semantic¹⁷⁴ information into account. We will see examples in the section on Blackfoot

the vertical line character U+007C is used. However, the presence of this character in morpheme shape or gloss transcriptions or in category names will cause issues since the system will treat it as a delimiter. In fact, the OLD employs a far lesser used character for this purpose, viz. the triple vertical bar delimiter U+2980.

¹⁷⁴That is, a phonology that parses surface representations into rich representations of morphemes and delimiters can take semantic information into account insofar as glosses are taken as representing meaning.

(§ 4.1) that make use of *rich lower* morphologies.

Morphologies that contain less information (i.e., those where both, or at least one, of their domains or ranges are impoverished) have the benefit of being significantly smaller and, as a result, quicker to compile. This is useful for expediting the task of debugging and developing morphophonologies, a task that may require numerous compilations. However, since a fully assembled morphological parser should return rich morpheme representations, those that contain morphologies (and hence morphophonologies) that analyze to impoverished representations must convert these to rich representations at parse time using the application logic.¹⁷⁵ The effect of this is that the time saved during the one-time compilation event is divided and offloaded onto the repeated parse event.

In general, production-ready morphologies should be *ruil* in order to allow for quick parse time and transparent phonologies. If compile time is intolerable during debugging/development, then an *iuil* morphology should be used provisionally, i.e., until parses are working as desired, at which point an equivalent *ruil* morphology should be swapped in. If phonological rewrite rules require reference to categorial and/or semantic information, then *rurl* morphologies should be used. And, of course, in this case it may help to use an *iurl* morphology during development.

To better illustrate all of this, consider the case of a very simple morphology that references a lexicon corpus containing the forms in (33) and a rules corpus containing those in (34).¹⁷⁶

¹⁷⁵That is, via Python as opposed to foma. Ambiguating sequences of morpheme shapes into sequences of shape-gloss-category triples via Python is slower than via foma.

¹⁷⁶In these examples, forms are, for brevity, represented as triples (m, g, c) , where m is

3.4. Morphologies

- (33) a. cheval, horse, N
b. chien, dog, N
- (34) a. le-s chat-s, the-PL cat-PL, NP
b. la tortue, the turtle, NP

Figures 3.7a, 3.7b, 3.7c, and 3.7d show the foma rewrite rule scripts that are generated for *rurl*, *ruil*, *iuil*, and *iurl* morphologies, respectively. All of these were generated using (33) as the lexicon corpus and (34) as the rules corpus.¹⁷⁷ All of these morphology scripts define four FSTs, one for determiners (DCat) one for nouns (NCat), one for number suffixes (PHICat), and one for the morphology (morphology). The morphology FST determines which sequences of morphemes and delimiters constitute licit words of the language.

the morpheme break, *g* is the morpheme gloss, and *c* is the category name.

¹⁷⁷The OLD allows morphology creators to decide whether or not to extract morphemes from the rules corpus via the aptly named `extract_morphemes_from_rules_corpus` attribute. In these examples, the morphemes are extracted from the rules corpus as well as the lexicon corpus. The effect here is that the determiners, plural suffix, and *chat* and *tortue* nouns are present in the example scripts. The rules extracted from the rules corpus are the sequences of category names and delimiters corresponding to the well-analyzed words in the corpus, i.e., the `morphology` FSTs in the scripts. (*Well analyzed* here refers to the fact that all morphemes in the analysis of a word are recognized, i.e., already present in the system as form objects.)

3.4. Morphologies

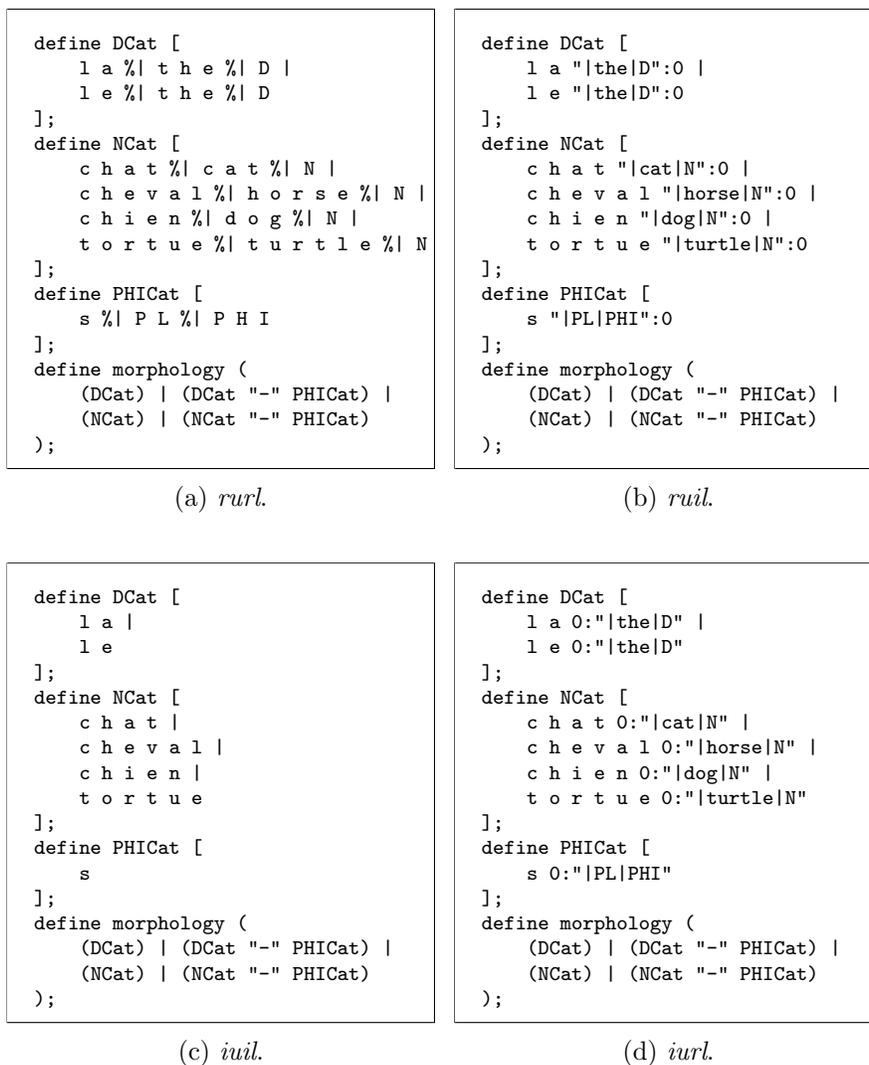


Figure 3.7: Morphology rewrite rule script examples.

The languages generated by the FST scripts in Figures 3.7a, 3.7b, 3.7c, and 3.7d can be captured by a restricted¹⁷⁸ CF phrase structure grammar.

¹⁷⁸These context-free (CF) grammars are restricted in that all rules are of the form

3.4. Morphologies

Figure 3.8 is a CF grammar that generates the same language as the *iutil* morphology FST defined in Figure 3.7c. The CF representation may facilitate understanding of morphology FSTs.

$S \rightarrow D$
 $S \rightarrow D-PHI$
 $S \rightarrow N$
 $S \rightarrow N-PHI$
 $D \rightarrow \{le, la\}$
 $PHI \rightarrow \{s\}$
 $N \rightarrow \{chat, chien, cheval, tortue\}$

Figure 3.8: Context-free (CF) grammar for a simple morphology.

The category FSTs defined in these scripts are specified as simple disjunctions of strings. Perhaps the simplest example is the **DCat** FST of Figure 3.7c, i.e., [1 a | 1 e] ;. This FST recognizes only the strings *le* and *la*; its network diagram is provided in Figure 3.9.

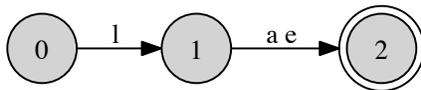


Figure 3.9: Network diagram for 1 a | 1 e.

The category FSTs for the morphologies that transform their inputs

$S \rightarrow (ND)^*N$ or $N \rightarrow T$, where N is the set of non-terminals (i.e., category names), D is the set of delimiters, and T is the set of terminal symbols (i.e., morphemes).

3.4. Morphologies

are a little more complicated. Consider the first disjunct of the DCat FST for the *iurl* morphology in Figure 3.7d, i.e., $1\ a\ 0:|\text{the}|D$. This defines three $\langle \text{upper}:\text{lower} \rangle$ symbol pairs: $\langle 1:1 \rangle$, $\langle a:a \rangle$, and $\langle 0:|\text{the}|D \rangle$. The last pair maps the empty symbol (0) on the upper side to the 6-character symbol $|\text{the}|D$ on the lower side.¹⁷⁹ Thus this morphology will generate $la|the|D$ from la and, conversely, will analyze $la|the|D$ to la . The network diagram for the determiner category FST of this *iurl* morphology is provided in Figure 3.10.

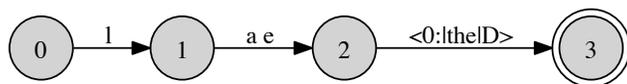


Figure 3.10: Network diagram for $1\ a\ 0:|\text{the}|D \mid 1\ e\ 0:|\text{the}|D$.

The DCat FST for the *rurl* morphology in Figure 3.7a should be relatively straightforward. The regular expression $1\ a\ \%|\text{t h e}|\ D$ recognizes a rich representation of the french determiner *la*. The vertical line character is a foma reserved symbol and is here escaped via prefixation of $\%$. The network diagram for the determiner category FST of this morphology is provided in Figure 3.11.

¹⁷⁹Multi-character symbols in foma regular expressions must be enclosed in double quotation marks.

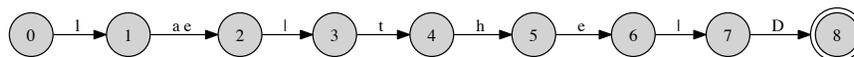


Figure 3.11: Network diagram for `l a %| t h e %| D | l e %| t h e %| D`.

When creating and generating an OLD morphology it is also possible to choose the value of the `script_type` attribute. The default value `regex` will generate morphology scripts like those in Figure 3.7, i.e., scripts that make use of the regular expression syntax. If the script type is set to `lexc`, however, then the morphology will be written in the *lexc* formalism (cf. Hulden, 2012; Beesley and Karttunen, 2003). The *lexc* equivalents of the *iuil* and *ruil* regex scripts in Figure 3.7c and Figure 3.7b are given in Figure 3.12a and Figure 3.12b, respectively. *Lexc* scripts are essentially right regular grammars. A line of the form `LEXICON Name` defines a non-terminal (NT) symbol *Name* on the left-hand side of the implicit rewrite rule. The right-hand sides of the rewrite rules are specified by the lines that follow, each of which is composed of a terminal symbol (i.e., morpheme or delimiter) optionally followed by a single NT symbol. (The number sign indicates that there is no non-terminal following the terminal.)

3.4. Morphologies

<pre>LEXICON Root Determiner ; DeterminerBreak ; Noun ; NounBreak ; LEXICON NounBreak chat BreakPhi; cheval BreakPhi; chien BreakPhi; tortue BreakPhi; LEXICON DeterminerBreak la BreakPhi; le BreakPhi; LEXICON BreakPhi - Phi; LEXICON Noun chat #; cheval #; chien #; tortue #; LEXICON Phi s #; LEXICON Determiner la #; le #;</pre>	<pre>LEXICON Root Determiner ; DeterminerBreak ; Noun ; NounBreak ; LEXICON NounBreak chat cat N:chat BreakPhi; cheval horse N:cheval BreakPhi; chien dog N:chien BreakPhi; tortue turtle N:tortue BreakPhi; LEXICON DeterminerBreak la the D:la BreakPhi; le the D:le BreakPhi; LEXICON BreakPhi - Phi; LEXICON Noun chat cat N:chat #; cheval horse N:cheval #; chien dog N:chien #; tortue turtle N:tortue #; LEXICON Phi s PL PHI:s #; LEXICON Determiner la the D:la #; le the D:le #;</pre>
--	---

(a) *iivil*.

(b) *rivil*.

Figure 3.12: Lexc morphology scripts.

Foma morphology scripts in the lexc formalism compile more quickly than those in the regex formalism. However, because of the way in which lexc scripts are automatically generated by the OLD (using UUIDs for lexicon names) they are not as readable as their regex equivalents.

3.5 Morphophonologies

An OLD morphophonology analyzes strings representing surface transcriptions into strings representing concatenations of morphemes. It is an FST constructed by composing a morphology FST and a phonology FST.

Within the OLD data structure, there is no dedicated morphophonology object. Instead, an OLD morphological parser object references existing phonology and morphology objects. When a *generate* request is issued against an OLD parser, the foma scripts of its phonology and morphology are merged into a single script wherein a morphophonology FST is defined as the composition of the morphology and phonology FSTs. A compile request on the parser results in the generation of a binary file representation of the morphophonology FST that can be used to perform apply up and apply down transformations. Just like the compile and generate requests of morphologies and phonologies, these requests can initiate long-running tasks and are thus delegated to worker threads; the requester must poll the parser until a change in the compile or generate attempt values has been registered, thus signifying that the attempt has terminated.^{180,181}

Once a parser has had its morphophonology FST generated and compiled, it can be used to analyze words into lists of candidate parses (via apply up), to generate surface representations from lists of parse representations (via apply down), and to return the most probable parse candidate for a

¹⁸⁰There are timeout limits for compile requests, which differ according to the object being compiled. Once the compile attempt exceeds the timeout its process is killed. These timeouts can be altered in `lib/utls.py`.

¹⁸¹Before a parser can be compiled, its morphophonology script must be generated, and that requires that the script of its morphology be generated as well. Note, however, that it is not necessary to compile either the phonology or the morphology of a parser.

3.5. Morphophonologies

given word transcription (via parse). Like a morphology, the compiled foma script of a parser may be downloaded by issuing a *serve compiled* request. In addition, the parser can be exported as a .zip archive which includes all the required files, Python modules, and a Python executable that makes it possible for a parser to be used locally.¹⁸²

Consider the simple French phonology in Figure 3.13. During generation, it transforms /la-s/ to ‘le-s’ (`plrlDetNeutralization`), /cheval-s/ to ‘chevaux’ (`alIrregularPlural`), removes delimiters (`delimiterDelete`), and defines a phonology FST that is the ordered composition of these three rules. Since this phonology maps morpheme shapes to surface realizations, it is compatible with any impoverished lower morphology. Figure 3.14 illustrates a morphophonology constructed by merging the *ruil* morphology of Figure 3.7b with the phonology in Figure 3.13.¹⁸³

```
define plrlDetNeutralization a -> e || .#. l _ "-" s .#. ;
define alIrregularPlural l "-" s -> u x || a _ .#. ;
define delimiterDelete "-" -> 0 ;
define phonology plrlDetNeutralization .o.
    alIrregularPlural .o.
    delimiterDelete ;
```

Figure 3.13: Toy French phonology.

¹⁸²Of course, in order to function locally, the local system must have foma and MITLM installed.

¹⁸³In fact, the OLD does not actually copy the morphology script into the morphophonology script. Instead, it inserts a foma `source /path/to/morphology/script` command which has the same effect. Since morphology scripts can be quite large, this strategy avoids unnecessary usage of disk space.

3.5. Morphophonologies

The morphophonology FST defined in Figure 3.14 recognizes all and only the words *la*, *le*, *les*, *chat*, *chats*, *cheval*, *chevaux*, *chien*, *chiens*, *tortue*, and *tortues*. It correctly analyzes (apply up) *chiens* to *chien|dog|N-s|PL|PHI* and *chevaux* to *cheval|horse|N-s|PL|PHI*.

```
define DCat [  
  l a "|the|D":0 |  
  l e "|the|D":0  
];  
define NCat [  
  c h a t "|cat|N":0 |  
  c h e v a l "|horse|N":0 |  
  c h i e n "|dog|N":0 |  
  t o r t u e "|turtle|N":0  
];  
define PHICat [  
  s "|PL|PHI":0  
];  
define morphology (  
  (DCat) | (DCat "-" PHICat) |  
  (NCat) | (NCat "-" PHICat)  
);  
define pluralDeterminerNeutralization a -> e || .#. l _ "-" s .#. ;  
define allIrregularPlural l "-" s -> u x || a _ .#. ;  
define delimiterDelete "-" -> 0 ;  
define morphophonology morphology .o.  
  pluralDeterminerNeutralization .o.  
  allIrregularPlural .o.  
  delimiterDelete ;
```

Figure 3.14: Toy French morphophonology.

The idea to model an entire morphophonology—i.e., a lexicon, morpho-tactic rules, and phonological transformations—as a single FST was originally proposed in Karttunen et al. (1992). This is an important development because it has the practical effect of significantly reducing the time com-

3.5. Morphophonologies

plexity of morphological analysis. In fact, in some cases analysis would not otherwise be possible. This is because phonological analysis can result in a large (sometimes infinite) set of candidates, i.e., sequences of phonemes and delimiters. Clearly, an independent morphological component tasked with filtering such a set down to only those candidates that correspond to valid sequences of morphemes would be slow in the best case and non-terminating in the worst.

To illustrate, consider that phonological analysis of *chevaux* using the phonology in Figure 3.13 will produce an infinite set of underlying representations. This is because the `delimiterDelete` rule will allow for any number of adjacent underlying delimiters, e.g., `ch---e--val-s`, `ch---e--vaux`, etc. Of course, we could eliminate the infinite ambiguity by refining the delimiter delete rule to more accurately reflect the distribution of delimiters, e.g., by requiring that they be surrounded by at least one non-delimiter on either side, i.e., `"-" -> 0 || "\" _ "\" ;`. This will result in a finite set of 80 candidates produced by phonological analysis of *chevaux*, e.g., `ch-ev-aux`, `ch-ev-au-x`, `ch-ev-a-ux`, `ch-ev-a-u-x`, `ch-ev-al-s`, `ch-evaux`, `ch-evau-x`, `ch-eva-ux`, `ch-eva-u-x`, etc. However, the fact remains that any reasonably complex phonology will produce during analysis sets of candidates that are too large for an independent morphology to parse in a reasonable amount of time.

However, when a morphology and a phonology are composed, the resulting morphophonology FST will not even consider phonological analyses that are morphologically invalid. This can be seen in the paucity of possible paths in the network diagram (Figure 3.15) for the morphophonology defined in

3.5. Morphophonologies

```
define DCat [  
  l a %| t h e %| D |  
  l e %| t h e %| D  
];  
define NCat [  
  c h a t %| c a t %| N |  
  c h e v a l %| h o r s e %| N |  
  c h i e n %| d o g %| N |  
  t o r t u e %| t u r t l e %| N  
];  
define PHICat [  
  s %| P L %| P H I  
];  
define morphology (  
  (DCat) | (DCat "-" PHICat) |  
  (NCat) | (NCat "-" PHICat)  
);  
define pluralDeterminerNeutralization [ a -> e ||  
  .#. l _ %| t h e %| D "-" s %| P L %| P H I .#. ] ;  
define impoverishMorphology [ %| \[ "-" | %| ]+ %| \[ "-" | %| ]+ -> 0 ||  
  _ [ "-" | .#. ] ] ;  
define allIrregularPlural l "-" s -> u x || a _ .#. ;  
define delimiterDelete "-" -> 0 ;  
define morphophonology morphology .o.  
  pluralDeterminerNeutralization .o.  
  impoverishMorphology .o.  
  allIrregularPlural .o.  
  delimiterDelete ;
```

Figure 3.16: Toy French morphophonology with *rurl* morphology.

Here the `pluralDeterminerNeutralization` rule has been given an even more restricted context, one that includes gloss and category information. That is, *a* becomes *e* only when it is the *a* in the determiner glossed “the” and only when it follows the plural suffix that is glossed “PL” and categorized as “PHI”.

The `impoverishMorphology` rule removes all gloss and category infor-

mation. The remaining rules (`alIrregularPlural` and `delimiterDelete`) can then once again perform transformations on underlying shapes only and are identical to their counterparts in Figure 3.14.

3.6 *N*-gram language models

A LM is a “probability distribution $p(s)$ over strings s that describes how often the string s occurs as a sentence in some domain of interest” (Chen and Goodman, 1999, p. 359). In the context of morphological parsing, a morpheme LM permits the assignment of probabilities to (and hence the ranking of) candidate parses, i.e., sequence of morphemes.¹⁸⁴ That is, when an OLD morphophonology returns multiple parse candidates, the LM can select the most likely candidate based on previously seen analyses. This section provides an overview of *N*-gram LM theory, including examples from morphological analysis and a description of the OLD and MITLM interfaces.

Consider the chain rule (Equation 3.1) for calculating the probability of an event m_1^n decomposable into n basic outcomes.¹⁸⁵

$$P(m_1^n) = \prod_{k=1}^n P(m_k | m_1^{k-1}) \quad (3.1)$$

¹⁸⁴The OLD does not include delimiters in its morpheme *N*-gram LMs. However, it might make sense to do so since some conventions for morphological analysis make use of more than one delimiter, e.g., cliticization via = and affixation via -. The effect of the current implementation is that the OLD will return the same probability to sequences of morphemes that differ only in their delimiters, e.g., *dog='s* and *dog-'s*. Since both analyses will be given equal probability, the system will not be able to rank them in any meaningful way.

¹⁸⁵In the notation used here, m_i is the i th element in a sequence and m_i^j denotes a sequence of elements i through j .

In the parsing context, the probability of a morphological analysis is the probability of the first morpheme, multiplied by the conditional probability of the second morpheme given its complete history (i.e., the first morpheme), multiplied by the conditional probability of the third morpheme given its complete history (i.e., the first and the second morphemes), etc. For example, according to the chain rule, the probability of the Blackfoot word in (35) is given in Equation 3.2.¹⁸⁶

- (35) *nitsiitsinoaayaawa*
 nit-iit-ino-aa-yi-aawa
 1-LOC-see-DIR-3PL-PRO
 AGRA-PREV-VAI-THM-AGRB-AGRB
 ‘I saw them’

$$\begin{aligned}
 P(\text{nit}, \text{iit}, \text{ino}, \text{aa}, \text{yi}, \text{aawa}) &= P(\text{nit}) \times \\
 &P(\text{iit}|\text{nit}) \times \\
 &P(\text{ino}|\text{nit}, \text{iit}) \times \\
 &P(\text{aa}|\text{nit}, \text{iit}, \text{ino}) \times \\
 &P(\text{yi}|\text{nit}, \text{iit}, \text{ino}, \text{aa}) \times \\
 &P(\text{aawa}|\text{nit}, \text{iit}, \text{ino}, \text{aa}, \text{yi})
 \end{aligned} \tag{3.2}$$

Under the *Markov assumption*,¹⁸⁷ one can approximate the conditional probability of an outcome given a minimal history, e.g.,

¹⁸⁶In these probability functions the shape of a morpheme should be understood as shorthand for a string representation of a (*shape, gloss, category*) triple.

¹⁸⁷There are LMs that are non-Markovian. In fact, the current state of the art in language modelling, i.e., stochastic memoizers (cf. Wood et al., 2009), involves estimating conditional probabilities given non-truncated histories.

$$\begin{aligned}
 P(\text{aawa}|\text{nit}, \text{iit}, \text{ino}, \text{aa}, \text{yi}) &\approx P(\text{aawa}|\text{yi}) \\
 &\approx P(\text{aawa}|\text{aa}, \text{yi}) \\
 &\approx P(\text{aawa}|\text{ino}, \text{aa}, \text{yi})
 \end{aligned}$$

This approximation is assumed under an N -gram LM. Specifically, it is assumed that the conditional probability of an element can be estimated given a history of $N - 1$ previously seen elements. Equation 3.3 provides the general method for computing the probability of a sequence given an N -gram LM.

$$P(m_1^n) = \prod_{k=1}^n P(m_k | m_{k-N+1}^{k-1}) \quad (3.3)$$

Thus, under 2-gram and 3-gram (i.e., bigram and trigram) models, the probability of (35), i.e., $P(\text{nit}, \text{iit}, \text{ino}, \text{aa}, \text{yi}, \text{aawa})$, is estimated according to expressions 3.4 and 3.5, respectively.¹⁸⁸

$$P(\text{nit}|\langle \text{s} \rangle)P(\text{iit}|\text{nit})P(\text{ino}|\text{iit})P(\text{aa}|\text{ino})P(\text{yi}|\text{aa})P(\text{aawa}|\text{yi})P(\langle / \text{s} \rangle|\text{aawa}) \quad (3.4)$$

$$P(\text{nit}|\langle \text{s} \rangle)P(\text{iit}|\langle \text{s} \rangle, \text{nit})P(\text{ino}|\text{nit}, \text{iit})P(\text{aa}|\text{iit}, \text{ino})P(\text{yi}|\text{ino}, \text{aa})P(\text{aawa}|\text{aa}, \text{yi})P(\langle / \text{s} \rangle|\text{yi}, \text{aawa}) \quad (3.5)$$

Given a lexicon of V morphemes, there are V^N logically possible N -

¹⁸⁸In order to create N -gram contexts for left edge morphemes and to generate a valid probability distribution it is necessary to prefix to each sequence one start and one end symbol, conventionally expressed by $\langle \text{s} \rangle$ and $\langle / \text{s} \rangle$, respectively (cf. Chen and Goodman, 1999). Note that this entails a history of only one element for estimating the conditional probability of a word-initial morpheme according to a trigram model, as opposed to the two-element history that might be expected (cf. expression 3.5). However, there is no point in doggedly pursuing consistency by prefixing $N - 1$ $\langle \text{s} \rangle$ elements to each sequence since $P(m|\langle \text{s} \rangle, \langle \text{s} \rangle) = P(m|\langle \text{s} \rangle)$.

grams; i.e., the sample space grows exponentially as the order N increases. By fixing N at some small value (usually 3), the Markov assumption built into N -gram LMs significantly reduces (relative to the chain rule) the amount of training data necessary in order to generate meaningful probability estimates for sequences of morphemes.

An intuitive method for estimating the conditional probability of a morpheme given a history of $N - 1$ previous morphemes is to count the number of times that the $(history, morpheme)$ sequence occurs in a given corpus (i.e., the frequency of the N -gram) and divide this number by the frequency of the history, cf. Equation 3.6 (Manning and Schütze, 1999). This approach is known as Maximum Likelihood Estimation (MLE) because it generates the model which assigns the highest probability to the training set, i.e., the corpus from which the counts are taken.

$$P_{MLE}(m_k | m_{k-N+1}^{k-1}) = \frac{C(m_{k-N+1}^k)}{C(m_{k-N+1}^{k-1})} \quad (3.6)$$

The problem with the MLE approach is that any $(history, morpheme)$ sequence that does not occur in the training corpus will result in a zero probability for the corresponding conditional probability. Since the probability of a word is the product of the conditional N -gram probabilities of its morphemes (cf. Equation 3.3), the MLE approach will result in many possible words incorrectly being evaluated as impossible (i.e., given a probability of 0). To illustrate, if the morpheme sequence *nit-iit-ino* never occurs in a particular training corpus of Blackfoot words, then, under a trigram

MLE LM generated from that corpus, $P(\text{ino}|\text{nit}, \text{iit}) = 0$ and therefore $P(\text{nit}, \text{iit}, \text{ino}, \text{aa}, \text{yi}, \text{aawa}) = 0$.

The solution to this problem is to redistribute some of the probability mass from the attested N -grams to the unseen ones. This is known as smoothing. The two main smoothing strategies are *backoff* and *interpolation*. Under backoff, one estimates a zero probability $P(m_k|m_{k-N+1}^{k-1})$ on the basis of the first lower order N -gram from which a probability can be generated. Under interpolation, the probability of a morpheme m_k is always a function of all of histories of length $\leq N - 1$ (Jurafsky and Martin, 2008, ch. 4).

In addition to specifying the smoothing algorithm, the model order N , and the training corpus, one can also specify a closed vocabulary for the LM. In the present context, a closed vocabulary is a finite set of morphemes that determines the set of possible N -grams for the LM. Specifying a closed vocabulary will result in MITLM a) ignoring all N -grams containing morphemes from outside of the vocabulary when the LM is generated and b) smoothing the N -gram probabilities over all words in the vocabulary (cf. MITLM, 2013). Since OLD morphologies (and hence morphophonologies) encode a finite lexicon of morphemes from which candidate parses can be constructed, the OLD interface allows for the lexicon of a morphology to be specified as the vocabulary for a LM.

The vocabulary of a LM can also be reduced in size, i.e., by clustering the elements in some way. Identification of morphemes with categories is an obvious clustering strategy. The benefit of category-based clustering is that it results in less data sparsity, i.e., fewer logically possible N -grams and, as

a result, fewer unattested *N*-grams. The downside, of course, is that there is less granularity in the model and, as a result, distinct parses with identical category sequences will be assigned identical probabilities.

A *generate* request on an OLD morpheme LM results in the creation of a number of files. First, the corpus of the LM is written to disk as a file where each line is a word that is represented as a space-delimited sequence of morphemes of the form *shape|gloss|category*. If the LM is specified as categorial, then the word lines are space-delimited sequences of category names. Second, if the LM object has a *vocabulary morphology* specified, then the lexicon of that morphology is used to write a vocabulary file which defines a closed vocabulary for the LM.

Once the corpus file and (optionally) the vocabulary file have been created, the OLD uses MITLM to count the *N*-grams present in the corpus and to estimate values for unseen *N*-grams. The order of the *N*-gram LM, i.e., the value of *N*, is determined by the value of its *order* attribute; the smoothing algorithm used to estimate unseen *N*-grams, one of a number of fixed options made available by MITLM,¹⁸⁹ is determined by the LM's *smoothing* attribute. The MITLM command executed by the OLD in order to estimate these *N*-gram values and write them to an ARPA-formatted¹⁹⁰ LM file is given in (36).

¹⁸⁹These are Maximum Likelihood (ML), Kneser-Ney (KN), modified KN (ModKN), and KN with “fixed discount parameters estimated from count statistics” (FixKN, Fix-ModKN) (cf. MITLM, 2013).

¹⁹⁰The ARPA format is described at <http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html>. The acronym references the U.S. Department of Defense Advanced Research Project Agency which sponsored the development of the format.

3.6. *N*-gram language models

```
(36) estimate-ngram -o <order> -s <smoothing> -t <corpusPath>
      -wl arpaPath (-v vocabularyPath)
```

Finally, the information encoded in the ARPA file is loaded into a Python data structure (an `LMTree` instance as defined by the `simplelm` module¹⁹¹) and is saved to disk as a Python pickle file.

Once an OLD morpheme LM has been generated, requesting *serve arpa* will return the ARPA file encoding it. A request to *get probabilities* will return a probability value for each morphologically analyzed word provided in the body of the request.

Consider an extremely simple corpus file created by repeating the word tokens in the rules corpus (34) from above 100 times. Such a corpus is represented by the four lines in Figure 3.17; i.e., the corpus file consists of these four lines repeated 100 times. The LM generated from this corpus file is represented by the ARPA file in Figure 3.18. This LM was created using MITLM’s `estimate-ngram` utility using an order of 3 (i.e., trigrams) and the Modified Kneser-Ney (ModKN) smoothing algorithm. (cf. Chen and Goodman, 1999; MITLM, 2013).

```
le|the|D s|PL|PHI
chat|cat|N s|PL|PHI
la|the|D
tortue|turtle|N
```

Figure 3.17: Simple LM corpus file (first four lines, repeated 100 times).

¹⁹¹<https://github.com/AdolfVonKleist/SimpleLM>.

3.6. N-gram language models

```
\data\  
ngram 1=7  
ngram 2=9  
ngram 3=6  
  
\1-grams:  
-0.979066 </s>  
-99 <s> -1.522879  
-0.847787 chat|cat|N -0.176091  
-0.847787 la|the|D -0.176091  
-0.847787 le|the|D -0.176091  
-0.485239 s|PL|PHI  
-0.847787 tortue|turtle|N -0.176091  
  
\2-grams:  
-0.607727 <s> chat|cat|N -1.522879  
-0.607727 <s> la|the|D -1.522879  
-0.607727 <s> le|the|D -1.522879  
-0.607727 <s> tortue|turtle|N -1.522879  
-0.258501 chat|cat|N s|PL|PHI -1.522879  
-0.394380 la|the|D </s>  
-0.258501 le|the|D s|PL|PHI -1.522879  
-0.979066 s|PL|PHI </s>  
-0.394380 tortue|turtle|N </s>  
  
\3-grams:  
-0.005884 <s> chat|cat|N s|PL|PHI  
-0.007845 <s> la|the|D </s>  
-0.005884 <s> le|the|D s|PL|PHI  
-0.007845 <s> tortue|turtle|N </s>  
-0.011821 chat|cat|N s|PL|PHI </s>  
-0.011821 le|the|D s|PL|PHI </s>  
\end\  

```

Figure 3.18: MITLM-generated ARPA LM file: *order* = 3, *smoothing* = *ModKN*.

Figure 3.18 tells us that there are seven unigrams (i.e., the five morpheme types in Figure 3.17 plus the start and end symbols `<s>` and `</s>`), nine

3.6. *N*-gram language models

bigrams, and six trigrams in the corpus. Under each `\N-grams:` header is a line for each *N*-gram which begins with its logarithm (base 10) conditional probability value,¹⁹² followed by the sequence of morphemes that constitutes the *N*-gram, followed optionally¹⁹³ by its \log_{10} backoff weight. The most probable unigram is `s|PL|PHI`, which makes sense since the plural suffix occurs 200 times in the corpus whereas all of the other morphemes occur only 100 times. The most probable trigrams are `<s> chat|cat|N s|PL|PHI` and `<s> le|the|D s|PL|PHI`.

The \log_{10} probability of the word *les* analyzed as *le-s* is -0.625432 (cf. Equation 3.7). Since this value is greater than the \log_{10} probability of the *la-s* analysis—i.e., -3.771002 (cf. Equation 3.8)—a morphological parser that disambiguates morphophonological analysis outputs via the LM in Figure 3.18 will rank *le-s* as more probable than *la-s*.¹⁹⁴

$$\begin{aligned} P(\mathbf{le}, \mathbf{s}) &= \log_{10}(P(\mathbf{le}|\langle \mathbf{s} \rangle)) + \log_{10}(P(\mathbf{s}|\langle \mathbf{s} \rangle, \mathbf{le})) + \log_{10}(P(\langle / \mathbf{s} \rangle|\mathbf{le}, \mathbf{s})) \\ &= -0.607727 + -0.005884 + -0.011821 \\ &= -0.625432 \end{aligned} \tag{3.7}$$

¹⁹²A probability $1 \geq p \geq 0$ in \log_{10} format (i.e., a *logprob*) is a real number $0 \geq r \geq -\infty$. For example, if $p = 0.0000567$, then $\log_{10}(p) = -4.246417$. Since $x \times y = 10^{\log_{10} x + \log_{10} y}$, sequences of conditional logprobs corresponding to sequences of morphemes can be summed (instead of multiplied) in order to calculate total word probability. This avoids arithmetic underflow. Note that the start symbol `<s>`'s $\log_{10}(p)$ value of -99 really denotes $-\infty$, i.e., a probability of 0.

¹⁹³Backoff weights only make sense for *N*-grams that are prefixes of larger *N*-grams.

¹⁹⁴This is actually an interesting case since a morphological parser of French should parse *les* as feminine (i.e., *la-s*) in some cases, e.g., in *les tortues*. In order to achieve this, it would be necessary both to build morpheme *N*-gram LMs across word boundaries and to morphologically parse phrases as a whole and not individually as words. However, the OLD does not currently support this. Though, even if this were supported, it would not be able to capture long-distance agreement effects like *J'ai vu des tortues. J'ai essayé de les attraper*.

$$\begin{aligned} P(\mathbf{1a}, \mathbf{s}) &= \log_{10}(P(\mathbf{1a}|\langle \mathbf{s} \rangle)) + \log_{10}(P(\mathbf{s}|\langle \mathbf{s} \rangle, \mathbf{1a})) + \log_{10}(P(\langle / \mathbf{s} \rangle | \mathbf{1a}, \mathbf{s})) \\ &= \log_{10}(P(\mathbf{1a}|\langle \mathbf{s} \rangle)) + \log_{10}(BW(\langle \mathbf{s} \rangle, \mathbf{1a})) + \log_{10}(BW(\mathbf{1a})) \\ &\quad + \log_{10}(P(\mathbf{s})) + \log_{10}(P(\langle / \mathbf{s} \rangle | \mathbf{s})) \\ &= -0.607727 + -1.522879 + -0.176091 + -0.485239 + -0.979066 \\ &= -3.771002 \end{aligned} \tag{3.8}$$

In computing the probability of the *la-s* analysis (Equation 3.8), we find that the LM does not define either $P(\mathbf{s}|\langle \mathbf{s} \rangle, \mathbf{1a})$ or $P(\langle / \mathbf{s} \rangle | \mathbf{1a}, \mathbf{s})$. That is, the corpus contains no instances of *la-s* at either the beginning or at the end of a word. $P(\mathbf{s}|\langle \mathbf{s} \rangle, \mathbf{1a})$ is therefore approximated as the backoff weight (*BW*) of the history $(\langle \mathbf{s} \rangle, \mathbf{1a})$ (i.e., $10^{-1.522879}$) multiplied by the conditional probability of the plural morpheme given the next longest history, i.e., $P(\mathbf{s}|\mathbf{1a})$. Since this probability is undefined as well, it is approximated as the backoff weight of the history $\mathbf{1a}$ (i.e., $10^{-0.176091}$) multiplied by the probability of the unigram \mathbf{s} (i.e., $10^{-0.485239}$). In the case of $P(\langle / \mathbf{s} \rangle | \mathbf{1a}, \mathbf{s})$, there is no backoff weight value for the history, but there is a probability for the word end marker given the next longest history (i.e., $P(\langle / \mathbf{s} \rangle | \mathbf{s})$) and it is this value (i.e., $10^{-0.979066}$) that is used to approximate the probability of the original event. The complete algorithm for computing the probability of a sequence given an LM in ARPA format is provided in Jurafsky and Martin (2008, ch. 4).

The reader should now have a sufficient understanding of *N*-gram LMs to understand how they are constructed and the role they play in OLD morphological parsers. Generally, poor parser performance can be traced to deficiencies in the morphophonology component (especially the phonology)

and not the LM component. That is, experience shows that when a morphophonology produces a set of candidate analyses that includes the correct one, the LM does an excellent job in selecting it. The OLD parser creator should do well simply by generating an LM based on a corpus that is as large as possible and whose morphological analyses accord with the types of parses that they are targeting.

3.7 Summary

The OLD morphological parser creator allows users to create multiple morphological parsers for various purposes. As described in the previous sections, users must a) supply ordered rewrite rules to build phonologies as FSTs and b) create OLD corpora from which the parser creator i) extracts lexica and morphotactic rules for the creation of morphology FSTs and ii) estimates N -gram LMs for the creation of parse candidate rankers. Since the parser creator is integrated into OLD fieldwork applications, it is able to use the data within those applications to expedite the otherwise tedious jobs of listing and categorizing lexical items, amassing grammatical category sequences, and collecting and properly formatting analyzed words for the estimation of N -gram LMs. The parser creator also facilitates the development of phonology scripts via functionality that allows for a test-driven approach. Finally, since the parsers created using the OLD morphological parser creator can be downloaded as fully functional Python-based command-line utilities, they can be used outside of the context of an OLD application.

The approach to creating morphological parsers presented here is a hy-

3.7. Summary

brid of expert-specified symbolic models and machine-induced statistical models. This approach is justified in the context of linguistic fieldwork on understudied languages (via an OLD application) because it allows researchers enough control over the mapping between morphological and surface representations so that analyses may be tested while not requiring the large data sets that would otherwise be needed for the construction of purely statistical models. Of course, the particular morphological parser creator described here is simply a particular application that has been added to the core OLD feature set; future work may include the incorporation of other parser creator applications which create different types of parsers.

The morphological parsers described here have clear potential to accelerate the creation of morphological analyses. This is a welcome feature in the context of fieldwork on endangered and under-studied languages where such data are relatively rare and where their rapid generation can help researchers to ask higher-level questions. In future work, the core components of OLD morphological parsers may also be co-opted for use in other tasks, e.g., the specification of transducers for conversion between orthographies.

Future work on the GUI for the OLD 1.0 web service will include graphical interfaces for creating parsers and interfaces that make use of these parsers to suggest analyses during data entry. This planned GUI will allow users to choose whether they want the system to suggest only the most probable parse or a list of parsers sorted in order of descending probability. Related to this, it would be interesting to discover whether the incorrect analyses returned by a parser are assigned, on average, probabilities that are lower than those of the correct analyses. If this is the case, then the in-

3.7. Summary

interface for configuring parsers to suggest candidate parses during data entry might allow users to calibrate the proposed parse suggester so that analyses with probabilities under a certain threshold would not be suggested.¹⁹⁵

This planned GUI will also allow users to test specified data sets (i.e., corpora) against specified grammatical models implemented as FSTs. The system will then return a report detailing the degree to which the data set is consistent with the model and will list forms that are not accounted for. This planned feature will help users to increase the consistency of their morpheme break and morpheme gloss values while also assisting in the development of more accurate grammatical models.

¹⁹⁵Thanks to Alexandre Bouchard-Côté for this suggestion.

Chapter 4

Blackfoot

This chapter provides a brief description of the Blackfoot language and details a Blackfoot data set that was created by combining the dictionary data (Frantz and Russell, 1995) with fieldwork data that were collaboratively compiled using the OLD software. The description of the data set in § 4.2 can be viewed as a real-world illustration of how the OLD can be, and has been, used, and the types of challenges that arise. This chapter gives context to chapter 5 *Morphological Parsers for Blackfoot* since it presents relevant facts about the language and describes the data set that is used to build and test the morphological parsers described therein.

4.1 Description

This section describes the Blackfoot language. It reviews some salient facts about its orthography, phonology, and morphology and provides context for the description of the data set in § 4.2 and the finite-state morphophonological analysis of the language presented in chapter 5.

Blackfoot is an Algonquian language spoken in Alberta (Canada) and Montana (USA). According to Lewis et al. (2013), it currently has some 3,500 speakers and is classified as “shifting”, meaning that “the child-bearing

4.1. Description

generation can use the language among themselves, but it is not being transmitted to children.” Four dialects are generally recognized (cf. Frantz and Russell, 1995): Kainai, Siksiká, Apatohsipikani, and Amskaapikani.

The Blackfoot phoneme inventory is given in Figure 4.1 (Elfner, 2006, p. 12) using International Phonetic Alphabet (IPA) symbols. The standard Blackfoot orthography (Frantz, 1978, 1991; Frantz and Russell, 1995)¹⁹⁶ is phonemic insofar as there is a graph (i.e., grapheme or digraph) for each phoneme in the language.¹⁹⁷ Figure 4.2 shows how the Blackfoot phonemes of Figure 4.1 are represented using the graphs of the orthography.

	labial	coronal	dorsal	glottal
plosives	p p:	t t:	k k:	ʔ
fricatives		s s:	x	
affricates		t͡s t͡s:	k͡s k͡s:	
nasals	m m:	n n:		
glides	w	j		

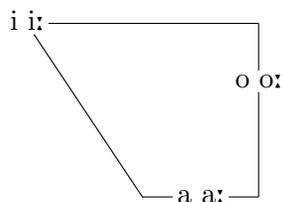


Figure 4.1: Blackfoot phoneme inventory (IPA).

¹⁹⁶Frantz (1978) argues for the principles that underlie the Blackfoot orthography; Frantz (1991, ch. 1) and Frantz and Russell (1995, appendix) delineate the graphs and their phonetic characteristics; Elfner (2006) contains a good summary of the orthography and an overview of the phonology of the language. Other relevant sources are Derrick (2006) and Denzer-King (2009).

¹⁹⁷“Each letter represents a distinctive sound of the language” (Frantz and Russell, 1995, p. 437).

4.1. Description

	labial	coronal	dorsal	glottal
plosives	p pp	t tt	k kk	'
fricatives		s ss	h	
affricates		ts tts	ks kks	
nasals	m mm	n nn		
glides	w	y		

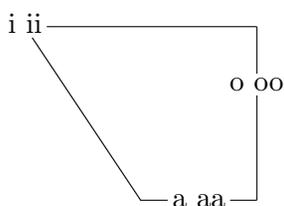


Figure 4.2: Blackfoot phoneme inventory (orthographic).

As a comparison of the two figures shows, all graphs of the orthography have their expected IPA values except that <y>¹⁹⁸ is /j/, <'> is /ʔ/, and <h> is /x/.¹⁹⁹ Length is phonemically distinctive in the language and long vowels and consonants are both indicated in the orthography by writing the graph twice and not by means of the triangular colon character ː, as prescribed by the IPA.

Morphemes—e.g., the morpheme break lines of the IGT examples in the grammar (Frantz, 1991) and the headwords of the dictionary (Frantz and Russell, 1995)—are transcribed *phonemically* using the orthography. However, orthographic transcriptions use this same inventory to express some

¹⁹⁸I follow the convention of enclosing graphs in angle brackets < and >.

¹⁹⁹Some sources (Silva, 2008) argue that there is no velar fricative phoneme but instead a series of preaspirated stops: /^hp/, /^ht/, /^hk/, /^hs/, /^hts/, and /^hks/. This is supported by the generalization that [x] never (or rarely) occurs in onset position and also by the fact that it invariably correlates with the devoicing of short vowels that precede it.

4.1. Description

level of *phonetic* detail. For example, the regular and predictable affrication of /t/ before /i/ is indicated in orthographic transcriptions (i.e., /t-i/ → <tsi>) whereas the similarly regular and predictable shortening of long vowels before long consonants (Frantz and Russell, 1995) is *not* (supposed to be) indicated in orthographic transcriptions.²⁰⁰

The phonological rules of the grammar (Frantz, 1991, pp. 152–155) can be seen as providing an explicit specification of which phonetic details orthographic transcriptions are supposed to encode.²⁰¹

Certain phonetic details *cannot* be represented orthographically. One example of this is the surfacing of the velar fricative /x/ as palatal [ç] before front vowels; the graph <h> is used in both cases. Similarly orthographically unrepresentable is the predictable devoicing of short vowels and the partial devoicing of long vowels before the velar fricative.

The phonology of the language produces several non-phonemic vowel phones that are expressed orthographically as digraphs. Figure 4.3 presents the phonetic vowel space of the language in IPA and, isomorphically, in the orthography. In closed syllables, short /a/, /i/, and /o/ are realized as [ə], [ɪ], and [ʊ], respectively; the /o/ phoneme varies freely between [o] and [u] (Elfner, 2006). The phoneme sequence /ai/ is realized as [eɪ], [æɪ], or [ɛɪ]²⁰² and as [ɛ] in closed syllables (Elfner, 2006). The sequence /ao/ is realized as

²⁰⁰Another predictable phonological alternation that is not indicated in the orthography is the glottalization of glides before voiceless vowels. A glottal stop is introduced between glides (and, in my experience with the language, nasals) and voiceless vowels (cf. Frantz, 1991, p. 18).

²⁰¹As a result, they might be more accurately characterized as *spelling* rules. However, given that the grammar discusses alternative rule formulations based on differences in pronunciation across speakers/dialects (cf. Frantz, 1991, p. 152, fn. 165) the *phonological* label is also accurate.

²⁰²Frantz and Russell (1995) attributes this variability to dialectal differences.

4.1. Description

[ɔ:] and as [ɔ] in closed syllables.

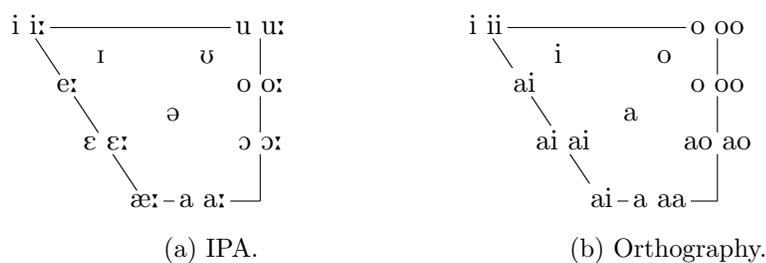


Figure 4.3: Blackfoot vowel phones.

In general, Blackfoot is morphologically polysynthetic and phonologically opaque. That is, words contain many morphemes and these are often hidden in phonetic form by general phonological transformations and allomorphy. This section briefly reviews the more salient aspects of the morphology and phonology of the language.

Inflected Blackfoot verbs are particularly morphologically complex. It is not uncommon for an inflected verb to contain seven or more morphemes, including but not limited to morphemes for tense, aspect, mood, negation, agreement, clause type, and adverbial meaning (Frantz, 1991). See Bliss (2013) for a detailed discussion of the morphology of Blackfoot verbs.

Blackfoot verbal morphemes are divided into four categories based on two parameters: transitivity and animacy (Frantz, 1991). Intransitive verbs with animate subjects are categorized as *animate intransitive* (*vai*) while those with inanimate subjects are categorized as *inanimate intransitive* (*vii*). Transitive verbs with animate objects are categorized as *transitive animate* (*vta*) while those with inanimate objects are categorized as *transitive inanimate*

4.1. Description

(*vti*). For a single core verbal meaning, Blackfoot may have a number of phonemically distinct verbs with different morphological properties, one for each transitivity/animacy possibility. In order to simplify this state of affairs, some sources (Armoskaite, 2011) argue for productive animacy-specific (in)transitivizing morphology *within* the verbal forms. However, the dictionary (Frantz and Russell, 1995), grammar (Frantz, 1991), and many other sources treat the *vai*, *vii*, *vta*, and *vti* forms as morphemes whose animacy- and transitivity-related properties must simply be memorized.

Inflected transitive animate (*vta*) verbs contain theme suffixes which indicate the grammatical role of the participants in the clause (Bliss, 2013; Frantz, 1991). Assuming a hierarchy of discourse participants where first person outranks second, second outranks third, and *proximate* third outranks an *obviative* third person, the *direct* theme suffixes signify that the more highly ranked participant is subject and the more lowly ranked one is object. The *inverse* theme suffixes, in contrast, signify the opposite assignment of grammatical roles to participants.

An inflected Blackfoot verb may contain between one and three agreement morphemes (as many as one prefix and two suffixes), each of which signifies some combination of person and/or number features. The shapes of these morpheme vary according to clause type, with Frantz (1991) identifying independent, conjunctive, subjunctive, imperative, and unreal clause types. Within each clause type, the forms and features of the agreement affixes vary according to the four-way transitivity/animacy distinction just described. Frantz (1991, pp. 147-151) provides comprehensive verb agreement paradigms.

4.1. Description

Each Blackfoot noun is grammatically specified as animate or inanimate and number suffixes agree with this animacy. In addition, animate singular nouns are marked as proximate or obviative by the shape of their number suffix (Frantz, 1991). Possession is indicated on nouns via affixes that are phonemically identical to a subset of the verbal agreement affixes. Certain nominal modifiers (i.e., morphemes that are often adjectives in other languages) may also be present as prefixes on nominal forms.

Blackfoot demonstratives also inflect for number, animacy, and proximity. In addition, different demonstratives indicate different literal or figurative proximity relative to a discourse participant (Frantz, 1991). See Schupbach (2013) for a recent analysis of Blackfoot demonstratives.

Certain syllables of Blackfoot words are phonetically prominent due to a relatively higher pitch during vowel production (Weber, 2013; Stacy, 2005; Van Der Mark, 2003). This phenomenon is known as *pitch accent* (Frantz, 1991) and is indicated orthographically via an acute accent on vowel characters. Particular vowels of particular morphemes are analyzed as bearing pitch accent underlyingly (Frantz and Russell, 1995), a phenomenon that may be labelled *lexical accent*. However, certain accented vowels are not attributable to lexical accent and recent work (Weber, 2013; Weber and Allen, 2012; Kaneko, 1999) has uncovered generalizations that predict where such non-lexical accent occurs, depending on the category of the word, the number of syllables, and the location of any lexical accent.

General phonological transformations, as well as transformations that are specific to a particular morphological domain or to a particular set of lexical items, may conspire to obscure the morphological composition of a

4.1. Description

Blackfoot word. An example of a general transformation is the affrication of /t/ before /i/ (Frantz, 1991, p. 154). Domain-specific alternations are exemplified by the surfacing of /o-a/ sequences as [a:], only when the /-a/ subsequence is not part of a suffix (Frantz, 1991, p. 152). Lexically conditioned phonological transformations are represented by the suffixation of /mm/ on a small set (ca. 50) of intransitive verbs (Frantz and Russell, 1995, p. xvii), only when a semivowel-initial agreement morpheme immediately follows (Frantz, 1991, p. 82). While the grammar (Frantz, 1991) characterizes such lexically conditioned transformations as allomorphy, it should be observed that i) the dictionary does not (at least not systematically or comprehensively) list these allomorphs and ii) it is standard practice for morphemic analyses (in the Blackfoot OLD, in the grammar (Frantz, 1991), and elsewhere) to make use of underlying phonemic representations of morphemes and not allomorphs. The phonological and allomorphic analyses of Frantz (1991) are reviewed in more detail in § 5.2.1 where they are implemented as finite-state transducers (FSTs).

The form in (37) illustrates the use of the orthography in both the orthographic transcription and morpheme break lines. Here we can see the effects of two phonological transformations—*t-Affrication* (*ti* → *tsi*) and *Postsibilization* (*sih* → *ss*) (Frantz, 1991)—as well as the introduction of non-lexical prominence, as indicated by the acute accent.

- (37) *Nítsspiyi.*
nit-ihpiyi
1-dance
'I danced.'

(Frantz, 1991, p. 28)

4.2. The Blackfoot data set

The morpheme categories of the dictionary are given in Table 4.1 (Frantz and Russell, 1995, pp. xii, 14–16). These categories are discussed further in § 4.2.1, which provides an overview of a corpus of morphemes defined in the Blackfoot data set.

abbreviation	category
<i>adt</i>	adjunct
<i>dem</i>	demonstrative
<i>fin</i>	final
<i>med</i>	medial
<i>nan</i>	animate noun
<i>nar</i>	animate relational noun
<i>nin</i>	inanimate noun
<i>nir</i>	inanimate relational noun
<i>pro</i>	pronoun
<i>und</i>	uninflected
<i>vai</i>	animate intransitive verb
<i>vii</i>	inanimate intransitive verb
<i>vta</i>	transitive animate verb
<i>vti</i>	transitive inanimate verb
<i>vert</i>	verb root

Table 4.1: Morpheme categories of the Blackfoot dictionary (Frantz and Russell, 1995).

4.2 The Blackfoot data set

This section describes *the Blackfoot data set*, a collection of 23,708 forms (cf. Table 4.2) that was created by taking the original fieldwork data from contributors to the Blackfoot OLD²⁰³ and adding to it the morphemes and

²⁰³<http://bla.onlinelinguisticdatabase.org>.

4.2. The Blackfoot data set

example words and sentences from the *Blackfoot Dictionary of Stems, Roots, and Affixes* (Frantz and Russell, 1995). Provided here are relevant statistics on this data set, an explanation of how its data were collected, an overview of a corpus of morphemes (§ 4.2.1) and a corpus of well analyzed words (§ 4.2.2) within the data set, and a review of some salient problems with and inconsistencies within it.

source	form count
original fieldwork	7,416
Blackfoot OLD grammar (Frantz, 1991)	1,020
other sources	467
dictionary (Frantz and Russell, 1995)	14,805
Total	23,708

Table 4.2: Sources of the Blackfoot data set.

The Blackfoot OLD has been in existence since September 2008. The data set described here contains the data in the Blackfoot OLD as of September 15, 2012 (the download date). At that time, the Blackfoot OLD contained 8,903 forms, of which 7,416 are from original fieldwork,²⁰⁴ 1,020 are from the *Blackfoot Grammar* (Frantz, 1991),²⁰⁵ and 467 are from other sources. The Blackfoot OLD data are, for the most part, sentences with morphological analyses. While a good number (1,928) of the forms in the Blackfoot OLD are from my own fieldwork, the majority were elicited and

²⁰⁴That is, 7,416 forms lack a specified source; these are, therefore, presumed to have been generated via original fieldwork.

²⁰⁵Thanks to Audra Vincent, Valerie Marshall, and Andy Matheson for their work on transcribing data from Frantz (1991) into the Blackfoot OLD.

4.2. The Blackfoot data set

entered by the other Blackfoot researchers who have contributed to the Blackfoot OLD. I would like to thank these contributors and their consultants²⁰⁶ for allowing me to use their data in this study.

In order to construct the data set described here, the data in the Blackfoot OLD at the download date were copied to a locally served OLD application²⁰⁷ and to this local database were added 14,805 forms extracted (via optical character recognition (OCR) and some post-processing) from the dictionary (Frantz and Russell, 1995). These consist of a large lexicon of morphemes²⁰⁸ (i.e., dictionary headwords) and the morphologically complex words and sentences that illustrate their usage.

In creating a lexical OLD form from each morpheme in the dictionary, the citation forms of the latter supplied the values for both the orthographic transcription and morpheme break attributes of the former. The translation of the dictionary headword supplied the values for both the morpheme gloss

²⁰⁶The original fieldwork data in the Blackfoot OLD come from three speakers (one from the Kainai reserve/dialect and the other two from the Siksiká), 15 fieldworkers, and 15 enterers. The speakers are BB, RE, and NB. The elicitors are Andrei Anghelescu, Solveiga Armoskaite, Michael Barrie, Heather Bliss, Strang Burton, Joel Dunham, Sara Johansson, Meagan Louie, Valerie Marshall, Kim Meadows, Maria Amélia Reis Silva, Elizabeth Ritter, Abigail Scott, Natalie Weber, and Martina Wiltschko. The OLD contributors who entered the data are Solveiga Armoskaite, Dylan Bandstra, Michael Barrie, Heather Bliss, Joel Dunham, Rebecca Hanson, Sara Johansson, Meagan Louie, Valerie Marshall, Andy Matheson, Kim Meadows, Maria Amélia Reis Silva, Abigail Scott, Audra Vincent, and Natalie Weber.

²⁰⁷Because of copyright issues, it was not possible to simply upload the data from the dictionary to the publicly served Blackfoot OLD application, even though the Blackfoot OLD is only accessible to some 50 registered users. This is why the data set described here was built by adding the dictionary data to a *local* copy of the data in the Blackfoot OLD.

²⁰⁸Some of the dictionary entries—in particular, the verbs, which some researchers (cf. Armoskaite, 2011) decompose into lexical and argument structure components—are arguably morphologically complex. However, since most researchers provide morphological analyses built upon morphemes that correspond to the entries of the dictionary, it is fine to treat these entries as morphemes.

and translation attributes of the OLD form. Where the dictionary translation contained multiple English words, all space characters were replaced by periods in order to arrive at a spaceless morpheme gloss string. The abbreviated category of each dictionary entry provided the OLD form's category value.

The example words and sentences of the dictionary (Frantz and Russell, 1995) are not morphologically analyzed in their source form and therefore the forms in the data set created from these have no morpheme break or morpheme gloss values. However, prior to the creation of the data set described here, a number of example words and sentences from the dictionary were morphologically analyzed and manually entered into the Blackfoot OLD.

In the majority of the forms in the data set, both the orthographic transcription and morpheme break values are written using the graphs (i.e., graphemes and digraphs) of the standard Blackfoot orthography (Frantz, 1978, 1991; Frantz and Russell, 1995). Following conventions established in the grammar (Frantz, 1991) and the dictionary (Frantz and Russell, 1995), the orthographic transcriptions of the data set encode a good deal of phonetic detail, whereas the morpheme break values are phonemic. See § 4.1 above for an overview of the orthography and the conventions of its use. However, see § 4.3.3 below for a discussion of how certain forms in the data set deviate from these orthographic conventions.

Table 4.3 lists the (morpho-syntactic) categories in use in the Blackfoot data set along with the number of forms that have each category. This category information is crucial in allowing us to divide the forms in the data set into a corpus of morphemes on the one hand and a corpus of morphologically

4.2. The Blackfoot data set

complex words, phrases, and sentences on the other. A particular corpus of morphemes is described in § 4.2.1. Within the corpus of morphologically complex forms it is possible to identify a corpus of morphologically well analyzed words and various subcorpora thereof; these corpora are described in § 4.2.2.

category	count	category	count
<i>none</i>	15,302	thm	10
sent	3,287	agra	10
vai	1,280	drt	7
nin	857	mod	7
nan	847	dem	6
vta	604	PN	6
adt	432	num	6
vti	408	prev	4
vii	305	pro	4
vrt	79	stp	4
oth	57	whq	4
fin	52	ten	3
nar	44	asp	3
agrb	28	nir	2
med	19	dim	1
und	18	o	1
Total	23,708		

Table 4.3: Forms in the Blackfoot data set, grouped by category.

4.2.1 Morphemes

This section describes a corpus of morphemes defined within the Blackfoot data set. This corpus is important because it is used to build the lexica of the morphological parsers described in chapter 5.

The corpus of morphemes is defined via the OLD search expression given in (38). (See § 2.3.6 for an explanation of the OLD search expression syntax.)

```
(38) ["and", [
      ["Form", "syntactic_category", "name", "in",
       ["nan", "nin", "nar", "nir", "vai", "vii",
        "vta", "vti", "vrt", "adt", "drt", "prev",
        "med", "fin", "oth", "o", "und", "pro",
        "asp", "ten", "mod", "agra", "agrb", "thm",
        "whq", "num", "stp", "PN"]],
      ["not", ["Form", "morpheme_break", "regex", "[ -]"]],
      ["not", ["Form", "id", "in", [15717, 23429]]]]]
```

This search expression returns 4,267 forms. Put simply, it returns all forms that i) have a morpheme category (lines 2-7), ii) do not contain a space or a hyphen in their morpheme break value (line 8), and iii) do not have an id of 15717 or 23429 (line 9).

Restriction iii) simply excludes the morphemes *á-* ‘DUR’ and *-hkayi* ‘certain’ from the corpus of morphemes. The ‘durative (DUR)’ verbal prefix *á-* is excluded because I wanted parsers created using this morpheme corpus as lexicon to return analyses containing the morpheme *á-* ‘imperfective (IMP)’, which is a morpheme that is phonemically and categorially identical to the ‘durative’ morpheme but which has a different gloss.²⁰⁹ The *-hkayi* ‘certain’ suffix (which affixes to demonstratives) was excluded because it is inappropriately categorized as *agra*, a category that is intended for the person prefixes on verbs and nouns (see below).²¹⁰

²⁰⁹The decision to exclude the ‘durative’ morpheme from the corpus of morphemes in favour of the ‘imperfective’ morpheme is based on my own analysis that the ‘imperfective’ label more accurately describes the semantics of the *a-* prefix. However, because many of the morphologically well analyzed words in the Blackfoot corpus use the ‘durative’ gloss in their analyses, the parsers discussed in chapter 5 might perform better if I were to allow both morphemes or just the ‘durative’ one in the corpus of morphemes. A better solution would be to alter the data set so that one of the two glosses is used consistently. I leave resolution of this issue for future work.

²¹⁰There are 16 forms in the data set which use the morpheme *-hkayi* ‘certain’ in their morphological analyses. Future work on improving the consistency of the data set should

Restriction ii) excludes from the corpus of morphemes all forms that contain spaces or hyphens in their morpheme break values. Such forms appear to be morphemes (since they have morpheme-level categories) yet the presence of spaces and/or hyphens indicates morphological complexity. When the morpheme break value contains a space, then there are multiple words in the form and the morpheme-level category is clearly inaccurate. However, when the morpheme break value contains one or more hyphens (and no spaces) then this indicates that a researcher has reanalyzed a morpheme from the dictionary as morphologically complex. An example of this is the form *ssoksi* which is a morpheme in Frantz and Russell (1995) meaning ‘be heavy in weight’ but which is reanalyzed in the Blackfoot data set as a verb root (*vrt*) *ssok* ‘heavy’ followed by a final (*fin*) *si* ‘be’. (See below for a discussion of the *vrt* and *fin* categories.)

Restriction i) causes the morphemes corpus to contain only forms that are categorized with one of the presumed morpheme-level categories. Table 4.3 above lists all of the categorization possibilities in the data set. The list of morpheme-level categories in lines 3-7 of (38) contains all of the possible categories except *sent*, *dem*, *dim*, and *none* (i.e., lack of a category). Forms that are categorized as sentences (i.e., via *sent*) are clearly not morphemes. Forms that lack a category are presumably not morphemes either. Even if there are category-less morphemic forms in the data set, these should be excluded from the corpus of morphemes because of the way that the OLD

include the assignment of a more appropriate category to this morpheme. In the meantime, parsers that use this morphemes corpus to generate their lexica will analyze demonstratives that end in *-hkayi* as containing the *stp*-categorized suffix *-hka* ‘invisible (INVS)’. See the discussion of the *stp* category below.

makes use of categories in auto-generating the category string value of forms. Recall from chapter 2 above that the OLD attempts to identify morphemes in the morphological analysis provided by a user and then uses the categories of the identified morphemes in order to generate a *category string* value for the form. Morphological analyses that reference a category-less morpheme will have ‘???’ in their category string values and will not be considered morphologically well analyzed (cf. § 4.2.2 below).

The demonstrative category (*dem*) is not considered to be a morpheme-level category for the purpose of defining the corpus of morphemes, cf. (38). This is because all of the forms categorized as *dem* in the data set (with the exception of *amo* ‘that’) are analyzed as morphologically complex. Instead of the *dem* forms, the corpus of morphemes contains the forms categorized as demonstrative roots (*drt*), cf. Taylor (1969). These are *ann*, *om*, *am*, *m*, and *n*. The demonstratives that contain the shapes *anno* and *amo* are assumed to be composed of a *drt*—i.e., *ann* and *am*, respectively—followed by the so-called “o-theme” suffix (cf. Taylor (1969, pp. 206–207) and Frantz (1991, p. 63, fn. 69)) which has shape *o*, is glossed as ‘PROX’, and “apparently references a position, actual or figurative, which is especially close to the speaker” (Taylor, 1969, p. 207). One consequence of the decision not to consider *dem* as a morpheme-level category is that parsers built using this morpheme corpus to generate their lexica will never return parses that contain the morpheme *amo* ‘that’; only analyses with *am-o* ‘DEM-PROX’ will be returned. Future work on defining morpheme corpora for the generation of parser lexica may allow for a greater degree of flexibility in the morphological complexity of words based demonstrative forms.

The diminutive category *dim* is only assigned to the suffix *-sst* ‘DIM’ which attaches to demonstrative roots, cf. Frantz (1991, fn. 69 p. 63) and Taylor (1969, pp. 206–7). Since, as a search reveals, this suffix is not used in any morphologically analyzed words in the data set, the *dim* category was excluded from the list of morpheme-level categories (cf. (38)) used to define the morphemes corpus.

A subset of the morpheme-level categories listed in (38) are from the dictionary (Frantz and Russell, 1995, pp. xii, 14–16); these dictionary-derived categories are listed in Table 4.1 above. The morpheme-level categories of (38) that are *not* from the dictionary are *oth*, *agrb*, *thm*, *agra*, *drt*, *mod*, *PN*, *num*, *prev*, *stp*, *whq*, *ten*, *asp*, and *o*. The remainder of this section summarizes the origins and use of these morpheme-level categories, beginning with the novel categories in the order provided above²¹¹ before moving on to the dictionary-derived ones.

The *oth* category (indicating “other”) is a catch-all used provisionally for morphemes whose category has not yet been ascertained.

The *agrb* category is given to the agreement suffixes on verbs and nouns which indicate person and number, e.g., *-wa* ‘third person singular’, *-yi* ‘third person plural’, *-hpinnaan* ‘first person plural’, *-hpoaawa* ‘second person plural’, *-o’pa* ‘first person inclusive plural’, etc. (cf. Frantz, 1991, pp. 147–151). Note that the data set under consideration here does not contain all of the extant Blackfoot person/number agreement suffixes; this is a deficiency that will be remedied in future work.

²¹¹The categories *drt* and *o* are not discussed below because they are already covered in the paragraph above which explains the omission of the *dem* category from the list of morpheme-level categories.

The theme (*thm*) category is assigned to the direct (*-o*, *-aa*, *-yii*) and inverse (*-ok*, *-okoo*, *-oki*) suffixes on *vta* verbs as well as to the “TI theme” suffixes (cf. Frantz, 1991, p. 44) (*-hp/-’p* and *-m*) on *vti* verbs. Refer to the description in § 4.1 above for further discussion of the direct/inverse and TI theme marking.²¹²

The *agra* category is assigned to the person prefixes on verbs and nouns, e.g., *nit-* ‘first person’, *kit-* ‘second person’, *ot-* ‘third person’, etc. (cf. Frantz, 1991, pp. 147–151).

The *mod* category is for the prefixes with modal semantics, e.g., *ááhkama’p-* ‘might’ and *ohkott-* ‘able’.

The *PN* category is used for proper nouns.

The *num* category is for the suffixes which mark number, in some cases proximate/obviative status, and which agree with the animacy of their host; these are *-iksi* ‘animate plural’, *-istsi* ‘inanimate plural’, *-wa* ‘(animate) proximate singular’, *-yi* ‘(animate) obviative singular’, *-yi* ‘inanimate singular’, and *-i* ‘non-particular’.

The preverb category (*prev*) is commonly used in the Algonquian literature (cf. Taylor (1969) for Blackfoot) and is used in the present data set to categorize a subset of the adjuncts of Frantz and Russell (1995), e.g., *it-* ‘LOC’.

The spatial/temporal proximity category *stp* is for a particular set of

²¹²There is also arguably an additional “mode” suffix between the *thm* and *agrb* suffixes which signifies the mood (or, in Algonquianist terminology, the order) of the verb, cf. Frantz (1991, ch. 19) and Muehlbauer (2005). These are the */-hs(i)/* suffix of the conjunct(ive) order and the */-hto/* suffix of the unreal order. (Déchaine and Wiltschko (2010) also posits */-hp/* as a suffix which signifies an independent mode.) The data set described here does not include morphemes for these mode suffixes.

suffixes on nouns and demonstratives (Taylor, 1969, p. 201), viz. *-ma* ‘stationary’, *-ka* ‘other time’, *-ya* ‘moving’, and *-hka* ‘invisible’ (Frantz, 1991, p. 66).

The *wh*-question category (*whq*) is used for the following interrogative morphemes (categorized as *und* in Frantz and Russell (1995)): *tsimá*, *takáá*, *tsa*, and *tahkáá*.

The *ten* category is assigned to the prefixes with tense-like semantics, viz. *áak-* ‘FUT’, *áyaak-* ‘FUT’, and *ii-* ‘PAST’.

The *asp* category is assigned to the prefixes with aspectual semantics, viz. *á-* ‘IMPF’ and *ákáá-* ‘PERF’.

The adjunct category *adt* is from the dictionary (Frantz and Russell, 1995) and is given to a large number of prefixes on nominal and verbal forms which, in general, have adjective-like and adverb-like meanings. Examples include *áka-* ‘old’, *ok-* ‘bad’, and *isimi-* ‘secretly’. The numerous negation morphemes of Blackfoot—viz. *miin-*, *min-*, *piin-*, *maat-*, *imaat-*, *káta-*—are also categorized as *adt*.²¹³

The final category *fin* is used to categorize suffixes “which must attach to other verb roots or stems, and which determine the category of the resultant stems” (Frantz and Russell, 1995, p. xv). The dictionary lists no more than two dozen morphemes with category *fin*, examples of which are *-wa’si* ‘become’ and *-imo* ‘have the odor of’. The Blackfoot OLD introduces a number of additional *fin*-categorized morphemes, including the so-called “pseudo-intransitivizer” suffix *-aaki* ‘PS.INTR’, cf. Frantz (1971, p. 46) and

²¹³For some reason the “elsewhere” negation morpheme *sa* (Frantz, 1991, ch. 16) is categorized as *oth* in the data set.

Taylor (1969, pp. 226–7). The morphemes categorized as finals in the data set do not appear to have the same distribution and more work is required in order to arrive at more accurate categorizations here (cf. § 4.3.1).

The medial category *med* is used to categorize morphemes “which must be incorporated into a verb or noun as a suffix” (Frantz and Russell, 1995, p. xv). The dictionary lists 15 medials, all but two of which (viz. *-ikim* ‘liquid’ and *-iksi* ‘wood’) reference body parts, e.g., *-sski* ‘face’ and *-ikinsst* ‘hand’.

The nominal categories are *nan*, *nar*, *nin*, and *nir*. The *nan* category is for grammatically animate nouns while the *nin* category is for inanimate ones, cf. the discussion of grammatical animacy in § 4.1 above. The *nar* and *nir* categories apply to the inherently relational animate and inanimate nouns, respectively; these nouns require a grammatical possessor (Frantz and Russell, 1995, p. xv).

The pronoun category *pro* is used in the dictionary to categorize “a very small set of items, including those built on the ‘pronominal base’ *iisto* such as *niistó* ‘I’, and *takáá* ‘who’” (Frantz and Russell, 1995, p. xv). In the data set discussed here, the “pronominal base” *iisto* is categorized as *pro* whereas forms like *niistó* ‘I’ and *kiistó* ‘you’ are analyzed as morphologically complex, i.e., as /n-iisto/ ‘1-PRO’ and /k-iisto/ ‘2-PRO’, respectively. The morpheme *takáá* ‘who’ is categorized as *whq*, as mentioned above. In addition, the distinct third person (DTP) verbal suffixes *-aistsi* and *-aiksi* (Frantz, 1991, cf.) are categorized as *pro*. Clearly the pronominal base morpheme and the DTP morphemes do not have the same distribution and further work on categorizing these morphemes is required.

The uninflected category *und* is used for the 15 “exclamations, such as *kika* ‘wait!’, and vocatives, such as *tsiki* ‘son!’ which take no affixes associated with nouns or verbs” (Frantz and Russell, 1995, p. xv).

The verbal categories are *vai*, *vii*, *vta*, and *vti*. These categories are for animate intransitive, inanimate intransitive, transitive animate, and transitive inanimate verbs, respectively. See the description in § 4.1 for a discussion of this four-way verb classification in Blackfoot.

Finally, the verb root category *vrt* is used for morphemes which “require a final to make up a verb stem” (Frantz and Russell, 1995, p. xv). That is, a verb root plus a final equals a form which behaves like a *vai*, *vii*, *vta*, or *vti* verb. An example is the *vrt o’t* ‘grasp’, which when suffixed by the final *-aaki* ‘PS.INTR’ results in the (*vai*-like) verbal form *o’taki* ‘take’.

4.2.2 Morphologically well analyzed words

This section discusses the morphologically well analyzed words in the Blackfoot data set and a number of corpora created from these. A word is morphologically well analyzed if it is broken down into a sequence of morphemes such that each morpheme exactly matches (i.e., in both shape and gloss) a morphemic form already in the data set (cf. § 2.2.4.5).²¹⁴ Such words are important for the creation and evaluation of Blackfoot morphological parsers, as described in chapter 5. This is because these forms have fully specified syntactic category string values and, as a result, we know the shape, gloss,

²¹⁴Recall that the French word /chien-s/ ‘dog-PL’ would be morphologically well analyzed *iff* it were in a database that also contained forms for /chien/ ‘dog’ and /s/ ‘PL’ and these forms had categories, e.g., ‘N’ and ‘Num’. In such a case, the well analyzed form would have a syntactic category string value of ‘N-Num.’

and category of each morpheme in the analysis of any such word. From a set of morphologically well analyzed words we can extract category sequences as morphotactic rules (cf. § 3.4), we can count length- N subsequences of fully specified morphemes for estimating N -gram language models (LMs) for candidate rankers (cf. § 3.6), and we can use these words as correct answers for evaluating how well a morphological parser performs.

In order to retrieve all forms in the Blackfoot data set that contain morphologically well analyzed words, an OLD search expression was created which returns all forms that are grammatical and have a syntactic category string value. This search result set was further filtered so that every form returned was either categorized as a sentence or had a word delimiter (i.e., a space) or a morpheme delimiter (i.e., a hyphen) in its syntactic category string value. That is, grammatical forms that represent sentences or which contain word/morpheme delimiters can be assumed to contain grammatical words.²¹⁵ Within each such word-containing form, the space characters were used to identify the words, where a word is defined here as a 4-tuple (t, b, g, c) consisting of a transcription t , a morpheme break b , a morpheme gloss g , and a category string c . In order to filter these down to only the *morphologically well analyzed* words, it was necessary to discard all words whose category

²¹⁵The search criteria just described will, in fact, also return forms which represent morphologically well analyzed *components* but which are not *words*, i.e., which are not forms that can occur on their own without additional inflectional morphology. An example of this is the *vta* verb *ikkahsimm* ‘find humorous’ which is analyzed as /ikkahs-imm/ ‘humorous-feel.toward’ *adt-vta* in the Blackfoot data set. This form cannot constitute a grammatical word without direct/inverse marking and person agreement morphology. A morphological parser whose morphotactics component is generated from a words corpus as defined here will, therefore, incorrectly recognize any *adt-vta* sequence as a grammatical word. Future work will refine the queries used to extract morphologically well analyzed words so that these false positives are filtered out.

4.2. The Blackfoot data set

string contained either ‘???’²¹⁶ or the sentential category *sent*.²¹⁷ As a result of this identification procedure, nearly every morphologically well analyzed word has a morphological analysis that contains only morphemes from the corpus of morphemes described in § 4.2.1 above.²¹⁸

	set	count
well analyzed word <i>tokens</i>		6,592
well analyzed word <i>types</i>		3,414
<i>gold standard</i> well analyzed word types		3,245

Table 4.4: Morphologically well analyzed word sets in the Blackfoot data set.

Table 4.4 lists the relevant word sets and their cardinalities. 6,592 morphologically well analyzed word tokens were identified using the procedure just described. From these well analyzed word tokens, it was possible to iden-

²¹⁶When generating syntactic category string values, the OLD uses ‘???’ as a dummy value for the category of morphemes that either cannot be identified or which have no specified category.

²¹⁷Since it is possible for a Blackfoot sentence to contain a single word analyzed as containing a single morpheme, there are words in the Blackfoot data set whose syntactic category string values contain *sent*. Therefore, it is necessary to filter out such words when defining a corpus of morphologically well analyzed words.

²¹⁸Because of the way in which morphemes and morphologically well analyzed words are identified, there are a few exceptions to this generalization which necessitate the use of the word *nearly*. As discussed in § 4.2.1 above, *dem* is not considered a morpheme-level category for the purpose of creating the morphemes corpus yet words analyzed as containing *dem*-categorized morphemes are present in the corpus of morphologically well analyzed words. Similarly, the morphemes *á-* ‘DUR’ and *-hkayi* ‘certain’ are excluded from the corpus of morphemes but may be present in the analyses of certain words in the corpus of morphologically well analyzed words. This means that a parser created using the morphemes corpus to generate its lexicon and the morphologically well analyzed words corpus to generate its morphotactics and LM ranker will be unable to correctly parse words whose analyses make use of the morphemes just discussed. While there are only a handful of morphologically well analyzed words that contain the morphemes under discussion, this is still an inconsistency that should be fixed in future work on these data sets.

tify 3,414 well analyzed word *types* and 3,245 *gold standard* well analyzed word types. These word sets are discussed below.

Given the above definition of a word as a 4-tuple (t, b, g, c) , any words comprised of identical 4-tuples are considered to be tokens of a single word type. Table 4.5 shows the 20 most common well analyzed word types.²¹⁹ Note that there are (at least) two well analyzed word types with the orthographic transcription *anna*. While both have identical morpheme break values, in one type the */-wa/* suffix is glossed as ‘PROX.SG’ and in the other it is glossed as ‘AN.SG’. Similarly, the demonstrative *ómi* is listed twice in Table 4.5 because its */-yi/* suffix is glossed as ‘OBV.SG’ in one type and as ‘IN.SG’ in another.

In order to facilitate evaluating the success of parsers, it is useful to have some method of choosing a single correct (i.e., *gold standard*) parse for those orthographic words, such as *anna* and *ómi*, which have multiple analyses. The method employed here is simply to choose the most common analysis for a given word in the data set, if one exists. Thus (*anna*, */ann-wa/*, ‘DEM-PROX.SG’, *drt-num*) is in the gold standard word set while (*anna*, */ann-wa/*, ‘DEM-AN.SG’, *drt-agrb*) is not. However, there is no word orthographically transcribed as *ómi* in the gold standard set since the two most common analyses both happen to occur 37 times. See § 4.3.2 for a discussion of how this method of choosing gold standard well analyzed words can be problematic. In order that the gold standard word set could easily be

²¹⁹Since nearly every DP in the Blackfoot contains a demonstrative—even DPs headed by proper nouns contain demonstratives—and since there are several types of demonstrative, each with multiple inflectional possibilities, it is not surprising that 16 of the 20 most common words are inflected demonstratives.

4.2. The Blackfoot data set

transcription	morpheme break	morpheme gloss	category string	count
ki	ki	and	und	265
na	ann-wa	DEM-PROX.SG	drt-num	153
matónni	matónni	yesterday	vii	137
anna	ann-wa	DEM-PROX.SG	drt-num	136
ni	ann-yi	DEM-OBV.SG	drt-num	117
anná	ann-wa	DEM-PROX.SG	drt-num	83
óma	om-wa	DEM-PROX.SG	drt-num	79
ámo	am-o	DEM-PROX	drt-o	76
anni	ann-yi	DEM-OBV.SG	drt-num	73
anna	ann-wa	DEM-AN.SG	drt-agrb	57
omiksi	om-iksi	DEM-AN.PL	drt-num	53
saahkómaapi	saahkómaapi	boy	nan	44
niksi	ann-iksi	DEM-AN.PL	drt-num	40
ómi	om-yi	DEM-OBV.SG	drt-num	37
ómi	om-yi	DEM-IN.SG	drt-num	37
ómiksi	om-iksi	DEM-AN.PL	drt-num	36
oma	om-wa	DEM-PROX.SG	drt-num	35
imitáá	imitáá	dog	nan	34
anní	ann-yi	DEM-OBV.SG	drt-num	33
omi	om-yi	DEM-IN.SG	drt-num	31

Table 4.5: 20 most common well analyzed words in the Blackfoot data set.

retrieved (e.g., as a corpus for Blackfoot parser construction, cf. chapter 5), a new OLD form object with the tag *gold* was created in the Blackfoot data set for each gold standard analyzed word and an OLD corpus was created which contains just these *gold*-tagged word forms.

Tables 4.6 and 4.7 show the ten most common verbals (i.e., verb-based words)²²⁰ and nominals (i.e., noun-based words), respectively, in the data set.²²¹ It is notable that the most common nominals all lack number (and

²²⁰With 137 occurrences, *matónni* ‘yesterday’ (vii) is by far the most common well analyzed verb-based word in the data set. This is because this word, which can function as a temporal modifier, was often used in elicitations that targeted temporo-aspectual interpretations of various verbal affixes.

²²¹A *verbal* in the present context is simply a word that contains a morpheme categorized as *vai*, *vii*, *vta*, *vti*, or *vrt*. Similarly, a *nominal* is a word that contains a morpheme

4.2. The Blackfoot data set

hence proximate/obviative) marking. This may be the result of word-final devoicing (Frantz, 1991, p. 5) which leads fieldworkers to assume that the *-wa* and *-yi* suffixes are not present. However, it is also possible that the fieldwork-generated data in the data set are from speakers who do not use the *-wa* suffix as widely as is asserted in Frantz (1991).²²²

transcription	morpheme break	morpheme gloss	category string	count
matónni	matónni	yesterday	vii	137
áyo'kaayaawa	á-yo'kaa-yi-aawa	IMPF-sleep-3PL-PRO	asp-vai-agrb-oth	7
nomohpooyi	n-omohp-ooyi	1-with-eat	agra-adt-vai	6
nitsooyi	nit-ooyi	1-eat	agra-vai	6
nitsinowa	nit-ino-wa	1-see-3SG	agra-vta-agrb	6
áyo'kaiksi	á-yo'kaa-iksi	IMPF-sleep-AN.PL	asp-vai-num	6
nitsínaan	nit-inaani	1-possess	agra-vai	5
nitsínoaa	nit-ino-aa	1-see-DIR	agra-vta-thm	5
kikatáí'nowaa	kit-káta'-ino-aa	2-INTER-see-DIR	agra-adt-vta-thm	5
iikssoksim	iik-ssok-im	INT-heavy-STAT.ANML	adt-vrt-fin	5

Table 4.6: Ten most common well analyzed *verbals* in the Blackfoot data set.

categorized as *nan*, *nin*, *nar*, or *nir*.

²²²Frantz (1991, p. 8) observes that “certain speakers omit the suffix *-wa* under as yet undetermined conditions. And many young speakers seem never to use it.”

4.2. The Blackfoot data set

transcription	morpheme break	morpheme gloss	category string	count
saahkómaapi	saahkómaapi	boy	nan	44
imitáá	imitáá	dog	nan	34
nínaa	nínaa	man	nan	31
pisátssaisski	pisátssaisski	flower	nin	30
pookáá	pookáá	child	nan	20
poos	poos	cat	nan	19
si'káán	si'káán	blanket	nan	18
otsitapíim	ot-atapíim	3-doll	agra-nan	18
ninna	nínaa	man	nan	16
aakii	aakíí	woman	nan	15

Table 4.7: Ten most common well analyzed *nominals* in the Blackfoot data set.

Finally, Table 4.8 shows the 20 most common category strings from among the 6,592 well analyzed word tokens in the data set. Most of these are nominal (i.e., *nan*, *nin*, *nir*, *nar*), demonstrative (*drt*) or verbal (i.e., *vai*, *vii*, *vta*, *vti*) morphemes with their expected inflectional morphology. However, see § 4.3.1 for a discussion of potentially problematic category sequences like *drt-agrb*.

4.3. Issues in the data set

category string	count
drt-num	1169
nan	348
und	335
nan-num	189
vii	152
nin-num	148
drt-agrb	143
agra-vai	132
nin	108
drt-o	99
adt-fin	89
drt-o-num	87
adt-vai	80
agra-nar-num	74
vrt-fin	71
drt	60
vai-agrb	59
asp-vai	58
agra-nar	57
agra-pro	56

Table 4.8: 20 most common category strings of well analyzed words in the Blackfoot data set.

4.3 Issues in the data set

Having described the Blackfoot language and a particular Blackfoot data set that was created using the OLD, the present section turns to a description of a number of salient issues with and inconsistencies within this data set. These issues involve the way that morphemes are categorized (§ 4.3.1), variability in how words are morphologically analyzed (§ 4.3.2), and inconsistencies in the use of the orthography (§ 4.3.3). These issues are illustrative of the types of challenges that arise when multiple contributors use the OLD software to amass and curate data on an under-documented language. In addition,

the issues described here serve as context for chapter 5 which describes two morphological parsers that were built and evaluated using the Blackfoot data set.

4.3.1 Morpheme categorization

As alluded to in § 4.2.1 above, there are a number of issues with the way that morphemes are categorized in the Blackfoot data set. The primary problem is that the categories are not consistently assigned to morphemes with the same distribution.

Consider that both the order (conjunct, unreal, etc.) of a verb form as well as its transitivity/animacy class (*vai*, *vta*, etc.) determine which person/number agreement affixes are permissible (Frantz, 1991, pp. 147–151). In addition, only certain combinations of person/number agreement prefixes and suffixes are grammatical. For example, *nitsspiyihpinnaan* /nit-ihpiyi-hpinnaan/ ‘1-dance-1PL’ *agra-vai-agrb* is grammatical while **nitsspiyih-poaawa* /nit-ihpiyi-hpoaawa/ ‘1-dance-2PL’ *agra-vai-agrb* is ungrammatical. This shows that the first and second person plural suffixes have different distributions despite both being categorized as *agrb*.

A more glaring example of miscategorization is the case of the morphemic form *wa* ‘AN.SG’ which is categorized as *agrb* in the data set.²²³ The *-wa* animacy/number agreement suffix on nouns and demonstratives patently does not have the same distribution as the person/number agreement suffixes on verbal forms (cf. Frantz, 1991).

²²³Forms like *anna* analyzed as /ann-wa/ ‘DEM-AN.SG’ explain the presence of the anomalous *drt-agrb* sequence in Table 4.8 above.

If one assumes, based on *agra-vai-agrb* forms like *nitsspiyihpinnaan* that any (*agra, vai, agrb*) sequence is grammatical, then ungrammatical words will be predicted. That is, a grammar based on such an assumption will overgenerate. As discussed in chapter 3, the morphological parser creator application of the OLD does make exactly this type of assumption when it generates FST-based morphologies from corpora. In general, grammatical models that overgenerate result in more candidate analyses for a parser to disambiguate but they do not categorically preclude a parser from returning a correct analysis. However, when such overgenerating models are used to accomplish inherently generative tasks—e.g., creating lists of grammatical words for, say, pronunciation dictionaries—the ungrammatical forms produced will be more problematic.

On the one hand, this type of morpheme miscategorization simply illustrates one of the challenges that arise when the OLD is used to store and curate data from under-documented languages. Since the OLD makes use of category information—e.g., in generating syntactic category strings and in building morphologies for parsers—OLD contributors have a motivation to think carefully about how they define and apply categories. In the specific case of the Blackfoot OLD, future work should involve the re-categorization of inflectional morphemes with the aim of reducing such overgeneration and, in general, more accurately capturing distributional patterns.

On the other hand, the miscategorization issue may be interpreted as indicating ways in which the OLD and the morphological parser creator (cf. chapter 3) could be improved. That is, a system which generates morpho-tactic models as simple morpheme category sequences may be unable to

accurately model the inflectional morphology of a language like Blackfoot without going to the extreme of assigning to every inflectional morpheme its own distinct category, an extreme which would eliminate the generalizing power that motivates the current design. It may be necessary to modify the OLD data structure so that it is possible to specify the combinatoric possibilities (i.e., subcategorization frames) for particular morphemes or morpheme categories, perhaps following the approach of FLE_x, cf. SIL International (2014a) and § 2.2.3 above.

Another approach to improving the accuracy of morphology FSTs for morphological parsers would be to allow users more control over the specification of morphological models. That is, if morphology writers could arbitrarily define morpheme classes, then distributional inaccuracies in the category-based classification could be sidestepped. Relevant here is the recent work of Snoek et al. (2014) which describes an approach to computationally modelling the morphology of another Algonquian language (Plains Cree) which involves using flag diacritics (Hulden, 2012) to handle long-distance morphological dependencies such as the circumfixal verb agreement morphemes discussed above. Future development may involve modifying the OLD to allow users more fine-grained control over morphology specification along these lines.

4.3.2 Variability in morphological analyses

There is a good deal of variability in how words are morphologically analyzed in the Blackfoot data set. There are a number of cases of orthographically identical words with distinct analyses. This reflects two distinct phenom-

ena. On the one hand, the ambiguity of natural language makes words with identical surface forms but different underlying structures an unavoidable empirical fact. On the other hand, this morphological ambiguity can in certain cases be traced back to inconsistencies in how morphemes are glossed as well as different assumptions about how to analyze the morphology of the language. In either case, inconsistencies in the data set with respect to morphological analysis can cause problems for parsers created using the OLD's morphological parser creator (chapter 3). This section describes this variability.

The inflected demonstrative *anna* provides an example of a variably analyzed word resulting from different analytical assumptions. While this word is segmented consistently as /ann-wa/, the /-wa/ suffix is glossed in some cases as proximate singular ('PROX.SG') and as animate singular ('AN.SG') in others.²²⁴ Blackfoot nouns of animate grammatical gender—and the demonstratives and modifiers that agree with them—may be marked either as proximate (i.e., local in some sense) via /-wa/, or as obviative (i.e., non-local) via /-yi/ (cf. § 4.1).²²⁵ Since /-wa/ can only occur in DPs headed by animate nouns,²²⁶ it is understandable that certain OLD contributors gloss it as animate singular.²²⁷ However, since it marks a proximal entity in contrast to a distal one, it also makes sense to gloss it as proximate.

Contrast *anna* with *ómi* (cf. Table 4.5). The *-yi* suffix here is either

²²⁴Yet another gloss for this suffix that is attested in the Blackfoot data set is '3SG'.

²²⁵Frantz (1991, p. 12) uses the terms *major* and *minor* third person for proximate and obviative, respectively, and uses *3* for major/proximate and *4* for minor/obviative.

²²⁶A further complication is that there is a /-wa/ suffix that marks third person on inflected verbs and which may or may not be the same as that on nouns.

²²⁷The proximate/obviative distinction is neutralized when plurality is involved; there is only one animate plural suffix: /-iksi/.

glossed as obviative singular (‘OBV.SG’) or as inanimate singular (‘IN.SG’). However, this is arguably a real ambiguity and not just a case of different glossing conventions. That is, *ómi* is the correct spelling of two homophonous demonstratives, a singular obviative animate one and a singular inanimate one.

Another example of variability of morphological analysis is related to the treatment of a phenomenon that is sometimes termed *initial change* and which involves a change in vowel quality at the left edge of the verb stem (cf. Frantz, 1991, pp. 35–38). Eventive predicates with a past tense interpretation and stative predicates with a present tense interpretation exhibit this phenomenon. It involves prefixation of /ii/ to consonant-initial stems and the replacement of stem-initial vowels with /ii/. However, the precise conditions under which initial change occurs have not been identified. In fact, it appears that the same predicate, with the same interpretation, can sometimes exhibit the phenomenon and sometimes not: compare (39a) to (39b). While (39) shows analyses which implicitly assume that the phonology is somehow responsible for the vowel change, certain Blackfoot OLD contributors posit an underlying morpheme with shape /ii/ in order to account for it, cf. (39c).

- (39) a. *Nitsúkska’si*.
nit-okska’si
1-run
‘I ran.’ (Frantz, 1991, p. 36), BLA OLD #20305

- b. *Nitókska'si*.
nit-okska'si
1-run
'I ran.' (Frantz, 1991, p. 36), BLA OLD #20304
- c. *Nitsúkska'si*.
nit-ii-okska'si
1-ic-run
'I ran.' BB, ML, April 1, 2010, BLA OLD #20305

In order to deal with morphologically ambiguous words when evaluating the performance of morphological parsers, the gold standard well analyzed word corpus was created (cf. § 4.2.2 above). Future work on the Blackfoot data set should seek to increase consistency in the analyses of words like *anna* and *nitsúkska'si*, i.e., cases where the inconsistency is due to different analytical assumptions. However, forms like *ómi* are genuinely morphologically ambiguous and future work may involve improving the morphological parser creator application so that the LM candidate ranker produced by means of it can take information outside of the word into context. For example, an obviative analysis of *ómi* would be ranked as optimal when the demonstrative forms a constituent with an animate noun and the inanimate analysis would be ranked optimal when the accompanying noun is inanimate.

4.3.3 Orthographic inconsistencies

The Blackfoot data set here is inconsistent in terms of how the orthography is used, both in the creation of orthographic transcriptions and in the creation of phonemic morpheme break values. Part of this inconsistency is due to the fact that the data set was created by amalgamating data from two

textual sources—the dictionary (Frantz and Russell, 1995) and the grammar (Frantz, 1991)—which make different orthographic assumptions. However, another portion of this inconsistency results from different Blackfoot OLD contributors using the orthography in different ways, in particular, when it comes to encoding phonetic details such as devoiced vowels and prominent (i.e., accented) ones.

A salient example of how the orthography is extended in order to transcribe phonetic detail in the Blackfoot data set involves the phenomenon of devoiced vowels, a phenomenon that the orthography (in its prescribed form) cannot encode. Short unaccented vowels are regularly devoiced before <h> (Frantz, 1991, p. 18), i.e., before /x/. The regularity of this alternation justifies the fact that the orthography does not provide a means of transcribing devoicing in this context. However, the dialectal and contextual factors that determine the occurrence of *word-final* devoicing (Frantz, 1991, p. 5) and “desonification” (i.e., the act of making a vowel soundless) (Gick et al., 2012) are not fully understood. It is therefore understandable that Blackfoot OLD contributors attempt to indicate word-final devoicing and/or desonification in orthographic transcriptions. Consider the Blackfoot OLD form (40).

- (40) *Nikéép(i)*.
 n-ikaa-ipi
 1-PERF-enter
 ‘I am inside now.’ (BB, NW, Jun 27, 2012)

Here the researcher (NW) parenthesizes the word-final <i>, presumably to indicate word-final devoicing of an inflected verb form as uttered by a

4.3. Issues in the data set

speaker (BB) of the Kainai dialect. This interpretation is corroborated by the researcher’s comment: “Frantz spells ‘enter’ with two i’s (*ipii*), but BB only devoices short vowels word-finally.” The Blackfoot OLD data set is full of such idiosyncratic attempts to convey this particular phonetic detail and this inconsistency is probably largely attributable to the lack of a *phonetic transcription* attribute for form objects in earlier versions of the database. I propose that word-final devoicing/desonification be omitted from the orthographic transcription values of the Blackfoot OLD and instead be specified in the phonetic transcription value via the combining ring below (U+0325) diacritic. For example, (40) should be phonetically transcribed as [niké:̥pi̥], to reflect this speaker’s actual pronunciation, and orthographically as <Nikaipii>,²²⁸ in accordance with the orthographic conventions.²²⁹

Example (40) illustrates another orthographic inconsistency involving the representation of phonetic detail.²³⁰ Adjacent lowercase epsilon symbols are being used here to indicate that the speaker produced a durationally long open-mid front unrounded vowel in syllable two. Orthographically, this should be represented as <ai>; however, the researcher is clearly striving for an explicit characterization of the vowel quality and duration here in

²²⁸Perhaps this form should be orthographically transcribed as <Nikaipi>, if, as indicated by the researcher, the phonemic shape of the ‘enter’ verb is /ipi/.

²²⁹One, perhaps arcane, aspect of Blackfoot phonetic transcription involves the transcription of partially devoiced long vowels. Orthographic sequences like <iihk> are long vowels that are initially voiced and become devoiced halfway through their pronunciation. The correct transcription here is [i̥:çk]. This is superior to both [i̥içk], which is problematic because the triangular colon is not being used for the long vowel, and [i̥:çk], which is potentially problematic because of the placement of the ring below the triangular colon.

²³⁰I must stress that I do not intend to single out or find fault with this particular fieldworker. The Blackfoot OLD was developed without any concerted attempt to enforce consistency in transcriptions or analytic representations. And, as stated earlier, previous versions of the system did not allow for distinct phonetic representations.

response to the ambiguity of the orthographic representation (cf. § 4.1).

Another common orthographic inconsistency in the data set is the transcription of prominence (pitch accent) via accented vowels. Many words in the data set have no prominence specification despite the fact it should be present.

Another notable orthographic discrepancy involves the use of so-called “breaking” /I/ in the morpheme break values of forms. Frantz (1971) and (variably) Frantz (1991) make use of /I/, a phoneme that is phonetically identical to /i/ but which causes postsibilization in a preceding /k/.²³¹ This is intended to account for the contrast in (41).

- (41) a. *Nitáakitsiniki*
 nit-yáak-itsiniki
 1-FUT-recount
 ‘I will tell a story.’ (Frantz, 1991, p. 31)
- b. *Kitáaksipii*
 kit-yáak-Ipii
 2-FUT-enter
 ‘You will enter.’ (Frantz, 1991, p. 31)

However, breaking /I/ is not represented in the dictionary (Frantz and Russell, 1995) nor is it widely used in the morpheme break values of the Blackfoot OLD. A morphophonology that parses orthographic transcriptions to sequences of morphemes drawn from the dictionary must therefore find some way of dealing with this fact. The approach taken in developing the phonology described in § 5.2.1 is to posit a rule that *optionally* breaks

²³¹Breaking /I/ is also intended to account for the presibilization of /t/ in certain morphemes in certain contexts. Thus /waanIt/ ‘say’ surfaces sometimes as [wa:nist] and sometimes as [wa:nik], cf. (Frantz, 1991, p. 152).

/k/ before /i/. A second approach would involve a phonology that is lexically aware such that it implements the /k/-breaking rule only when certain morphemes are involved, viz., those that should be transcribed with /I/. A third option would be to modify (i.e., fix) the lexical entries in the data set that are taken from Frantz and Russell (1995) so that breaking /I/ is represented.²³² A final possibility is that the /I/ analysis is incorrect and there is a more general morphological pattern that determines whether a /k/ will become an affricate/assibilant before /i/. Armoskaite (2006) proposes that /k/ assibilates before /i/ when the sequence occurs in a particular morphological domain that can be roughly characterized as the verb stem plus its prefixes. While this is a tempting possibility, data like (41a) would appear to constitute problematic counterexamples.²³³

4.4 Summary

This chapter has provided a brief description of the Blackfoot language (§ 4.1), has reviewed a Blackfoot data set created by combining data from the Blackfoot OLD with data from a widely used dictionary (Frantz and Russell, 1995) and grammar (Frantz, 1991) of the language, and has pointed out some salient issues and inconsistencies in this data set. The information

²³²Modification of the lexical entries of (Frantz and Russell, 1995) to indicate breaking /I/ is a task that could be partially automated but which would still require a considerable number of man-hours. Even so, the second option—i.e., specifying lexical conditions on the breaking of /k/ in the phonology—would require the same time-consuming identification of the relevant morphemes.

²³³However, if the morphological domain generalization of Armoskaite (2006) can predict the majority of breaking environments, then it may be less costly (i.e., than the previously discussed options) to encode in the phonology both this generalization and its putatively small number of lexical exceptions.

4.4. *Summary*

provided here should be useful both as an illustration of how the OLD has been used in an actual collaborative fieldwork project on an endangered language and as a reference for the data sets used in creating and evaluating the morphological parsers that are presented in chapter 5.

Chapter 5

Morphological Parsers for Blackfoot

This section discusses the results of using the morphological parser creator (chapter 3) to build parsers for the Blackfoot language using *the Blackfoot data set* (cf. § 4.2), i.e., data from the Blackfoot OLD,²³⁴ the dictionary (Frantz and Russell, 1995), and the grammar (Frantz, 1991). The best parser produced—which scores 17% on parse accuracy and has an F-score of 0.4—is presented as evidence of the utility of the parser creator and its potential to i) expedite data entry and analysis generation and ii) allow for the computational implementation and evaluation of theoretical linguistic models. The morphophonology of the parsers (as described in § 5.2.1 and § 5.2.2) is an explicit formulation of the phonological and allomorphic analyses of the grammar (Frantz, 1991) and the lexical analysis of the dictionary (Frantz and Russell, 1995).²³⁵ A small contribution to Blackfoot theoretical analysis is the explicit ordering of phonological rules (cf. Figure 5.1) which was discovered in the course of formulating the phonologies of these parsers.

²³⁴<http://bla.onlinelinguisticdatabase.org>.

²³⁵The full morphological analysis of the grammar (Frantz, 1991) is not modelled in any meaningful way since the morphotactic generalizations and inflectional paradigms are not captured by the morphologies of these parsers (cf. § 5.2.2).

Two parsers were created and then evaluated against randomly selected unseen test sets drawn from the gold standard morphologically well analyzed words of the Blackfoot data set (cf. § 4.2.2). Both parsers use the morphemes of the data set (§ 4.2.1) as lexicon and extract morphotactic rules (i.e., valid category sequences) and N -gram frequencies for language models (LMs) from the morphologically well analyzed words in the data set (§ 4.2.2). They differ in their phonologies. Parser 1 (§ 5.2) strives for faithfulness to the phonological transformations and allomorphic alternations of the grammar (Frantz, 1991). Parser 2 (§ 5.3) modifies the phonology of Parser 1 to recognize more surface-underlying mappings by removing vocalic prominence distinctions and segment length distinctions in surface transcriptions during generation.

The phonology of Parser 1 is presented in detail (§ 5.2.1) in order to show how a variety of (sometimes lexically conditioned) phonological transformations can be implemented using the finite-state transducer (FST) regular expression rewrite rule language. This description should also be of interest to scholars of Blackfoot since it is an explicit interpretation and formulation of a widely used morphophonological analysis of the language.

Parser 1 achieves a parse success rate of 0.14 and an F-score of 0.32. Parser 2 improves upon this with a parse success rate of 0.17 and an F-score of 0.4. With reference to the issues in the data set (cf. § 4.3), § 5.4 discusses the implications and future uses of the parsers and possibilities for their improvement.

5.1 Parser creation and evaluation procedures

While chapter 3 explains how the morphological parser creator application works in general, the present section explains how that application was used to build the two Blackfoot parsers described in the sections that follow. The methods used to evaluate the performance of these parsers are also described here.

A custom-built Python module—`old-parser-research`²³⁶—was created to facilitate the creation of parsers via the RESTful API of a live OLD application. This involves issuing the appropriate requests to build a phonology, a morphology, a LM, and a parser that references these three components. While the phonology is essentially just a foma script containing ordered CS rewrite rules, the creation of morphologies and LMs involves the creation of corpora based on specific searches (cf. chapter 3). The research module simply removes some of the complexity of issuing these requests and performs local caching of results in order to avoid redundant and time-consuming computations.

Once a parser has been created, the research module issues an *export* request in order to download all of its components to the local system (§ 3.5). This local copy of the parser is evaluated²³⁷ against the gold standard well analyzed words of the data set. The evaluation results in files detailing where

²³⁶The OLD parser research module can be downloaded from <https://github.com/jrwdunham/old-parser-research>. While it is certainly not required in order to create and test OLD morphological parsers, it is a convenient helper in this task. It also contains examples of code that interacts with a live OLD application using the RESTful API.

²³⁷The Python API of a local copy of the parser is used for evaluation simply to avoid the performance overhead of making HTTP requests to the RESTful API of a live OLD application.

the parser has failed as well as a set of performance measurements, including F-score.

An F-score (a.k.a. F-measure) (van Rijsbergen, 1975) is a standard metric for evaluating parsers and other NLP tools (cf. Jurafsky and Martin, 2008). More fine-grained than a simple measurement of parse accuracy, i.e., the number of correct parses divided by the number of parse attempts, the F-score is a function (5.3) of the parser's precision P (5.1) and its recall R (5.2) (cf. Goldsmith, 2001).²³⁸

$$P = \frac{\text{number of correct proposed morphemes}}{\text{number of proposed morphemes}} \quad (5.1)$$

$$R = \frac{\text{number of correct proposed morphemes}}{\text{number of correct morphemes}} \quad (5.2)$$

$$\text{F-score} = \frac{2PR}{P + R} \quad (5.3)$$

An intuitive illustration of precision and recall can be had by imagining a multiple choice test. The student who selects every option has perfect recall but terrible precision. The student who correctly answers only the one answer he knows has perfect precision but terrible recall.

To see how the precision and recall of parsers were calculated, compare the evaluation of three different parsers tasked with analyzing the Blackfoot word *nitsiitsinoaayaawa* where the gold standard parse is *nit-iit-ino-aa-yi-*

²³⁸The F-score function given in Equation 5.3 is actually the weighted harmonic mean of precision and recall where precision and recall are equally balanced (cf. Jurafsky and Martin, 2008).

aawa, cf. (42) repeated from (35).

- (42) *nitsiitsinoayaawa*
 nit-iit-ino-aa-yi-aawa
 1-LOC-see-DIR-3PL-PRO
 AGRA-PREV-VAI-THM-AGRB-AGRB
 ‘I saw them’

Parser 1 returns the correct analysis and thus has 1 for its accuracy, precision, recall, and F-score values. Parser 2 is exact (everything it guesses is correct) but incomplete (it misses some of the correct morphemes); it returns *nit-ino-aa-yi* and thus has a precision of $4/4 = 1$, a recall of $4/6 = 0.666$, and an F-score of $4/5 = 0.8$. Parser 3, on the other hand, is imprecise (it returns many false positives) but complete (it contains all possible correct morphemes); it returns *n-nit-yi-iit-yi-ino-wa-aa-yi-yii-aawa-a* and thus has a recall of $6/6 = 1$, a precision of $6/12 = 1/2$, and an F-score of $2/3 = 0.666$.

In addition to overall accuracy, precision, recall, and F-score, the OLD parser research module can compute independent measurements of the accuracy of the phonology, the morphophonology, and the LM. The accuracy of the phonology is the number of analyzed words where the parser’s phonology generates the surface representation from the gold standard underlying representation. If the phonology fails to do this, then the morphophonology and the LM will both, by necessity, fail also. When calculating phonological accuracy, the research module will also create a file that lists the specific underlying-surface pairs that the phonology was unable to account for. This is a useful data set since it can be used to improve the phonology.

The accuracy of the morphophonology is the number of instances wherein

the set of candidates output by the morphophonology FST contains the correct parse, divided by the number of parse attempts. The accuracy of the LM is the number of correct parses divided by the number of morphophonological successes. that is, LM accuracy indicates how well the LM is doing in those cases where it has a chance of choosing the correct parse. Note that the measure of morphophonological accuracy (and the measure of LM accuracy calculated on the basis of it) is estimated based on parse attempts over small sub-corpora. The reason for this is that a morphophonology may produce such large numbers of candidate parses—e.g., on average 1,600 per parse attempt for Parser 2—that caching all of these and calculating accuracy measures based on them results in a prohibitively time-consuming computation.

5.2 Parser 1

This section describes Parser 1. Its phonology (§ 5.2.1) is an ordered list of rewrite rules that adhere as faithfully as possible to the phonological and allomorphic analyses of the grammar (Frantz, 1991), cf. § 4.1. Its lexicon (§ 5.2.2) is extracted from the corpus of morphemes described in § 4.2.1, i.e., an OLD corpus of monomorphemic forms which, for the most part, come from the dictionary (Frantz and Russell, 1995). Its morphotactics (§ 5.2.2) is extracted from the category sequences of the corpus of morphologically well analyzed words in the Blackfoot data set, as described in § 4.2.2. Its LM (§ 5.2.3) is trained and tested on randomly generated disjoint sets drawn from the corpus of gold standard morphologically well analyzed words (cf.

§ 4.2.2). Parser 1 has a 14% success rate and an F-score of 0.32; these results are detailed and discussed in § 5.2.4.

5.2.1 Phonology

The phonology of Parser 1 is an FST rewrite rule script²³⁹ that is highly faithful to the phonological and allomorphic analyses of the grammar (Frantz, 1991), specifically, the 25 phonological rules listed in *Appendix B* and the lexically conditioned phonological alternations of *Chapter 15: Allomorphy* of that text. This phonology script includes nearly all²⁴⁰ of the relevant illustrative examples of the grammar (Frantz, 1991) as tests (cf. § 3.3); running the parser creator’s *perform tests* request confirms that the phonology passes all of these tests, thus showing that this phonology captures the phonological and allomorphic generalizations of the analyses in the grammar.²⁴¹

This section describes the phonology of Parser 1 in explicit detail, showing exactly how and why it differs from the analyses of the grammar (Frantz, 1991). Since it includes explicit formulations of all relevant rules as well as

²³⁹The complete foma phonology script described here can be found on the GitHub repository of the `old-parser-research` tool, i.e., at <https://github.com/jrwdunham/old-parser-research>; see the file `blackfoot_phonology_frantz91.script` in the `resources/` directory. By downloading this script, readers with foma installed can verify that the phonology behaves as described here.

²⁴⁰All of the examples from the phonological rules appendix of the grammar are included as tests in this phonology script. Certain tests from the allomorphy chapter are omitted because they are superfluous, not because the phonology cannot account for them. The tests were, in a few cases, minimally modified so that the morphemes in the morpheme break value would be consistent with the dictionary (Frantz and Russell, 1995) and so that non-lexical prominence effects—which the current phonology makes no attempt to account for—could be neutralized.

²⁴¹As explained in § 3.3, a test is a pair consisting of an underlying representation UR and a surface representation SR . If a phonology generates SR from UR , it passes the test. That is, the phonology passes even if it overgenerates, i.e., even if it generates additional surface representations $SR_1 \dots SR_n$ from UR .

their ordering (something which is *not* provided in Frantz (1991)), it is a small contribution to the theoretical linguistic analysis of the phonology and allomorphy of Blackfoot. To the extent that this phonology is faithful to these analyses, the evaluation of its performance is a measure of the observational adequacy (Chomsky, 1965) of said analyses.

When tested against the set of 3,245 gold standard well analyzed words (cf. § 4.2.2) the phonology is 21% accurate.²⁴² Inspection of the gold standard words that this phonology fails to account for reveals that, while issues with the test set (e.g., too much phonetic detail in orthographic transcriptions, cf. § 4.3) *are* a significant factor, the inability of the phonology to account for the location of non-lexical accent marking is a major problem. That is, when all prominence contrasts were removed in the gold standard test set (i.e., when all <á>, <í>, and <ó> graphemes were replaced with <a>, <i>, and <o>, respectively) the accuracy of the phonology increased to 40%. This finding reinforces the conclusion—addressed in works such as Weber (2013), Weber and Allen (2012), and Kaneko (1999)—that an analysis of non-lexical pitch accent marking is a major gap in the phonology of the grammar (Frantz, 1991).²⁴³

This section first lays out the ordering of general phonological rules, then presents and discusses first the general rules and then the lexically conditioned ones which capture the allomorphic alternations of the grammar

²⁴²This means that any parser built on top of this phonology can itself have at most a 21% success rate.

²⁴³Actually, Frantz (1991, pp. 90–91) does discuss a certain phonological generalization which introduces a specific type of non-lexical pitch accent in certain contexts: a falling accent is introduced where vowels meet at a morpheme boundary. Future work on the phonology described here will involve assessing and/or incorporating this generalization.

(Frantz, 1991).

Frantz (1991, p. 155) does not specify a complete ordering for the 25 phonological rules of the appendix, but offers instead the following summary.²⁴⁴

Rules apply for maximal “feeding” and minimal “bleeding” except that coalescence bleeds desyllabification, presibilization, and semivowel loss; i-absorption bleeds desyllabification, and gemination bleeds s-insertion. Desyllabification bleeds accent spread, but does not bleed nor feed any other rules.

Figure 5.1 contains a portion of the definition of the **phonology** FST of Parser 1’s phonology. This figure shows the ordered composition of 24 rewrite rules based on the phonological rules appendix of Frantz (1991). This ordering accords with the ordering generalizations made in the quote above. Coalescence comes before, and can therefore bleed, desyllabification, presibilization, and semivowel loss. The i-absorption rule precedes and therefore bleeds desyllabification, the gemination rule precedes and therefore bleeds s-insertion, and the desyllabification rule precedes and therefore bleeds accent spread. The application of desyllabification cannot possibly bleed any of the rules that follow it in the ordering of Figure 5.1.

While there are 25 rules listed in Frantz (1991, pp. 152–155), there are only 24 FSTs (implementing rules) in Figure 5.1. This discrepancy is due to the fact that 1) the s-insertion and neutralization rules are not included here and 2) a new rule is introduced (*break delete*) which elides the morpheme

²⁴⁴For readability, I replace the rule numbers of Frantz (1991) with the names of the rules.

delimiters (cf. Figure 5.25). The s-insertion rule ($\emptyset \rightarrow s \mid I _ t$) is not included here since the relevant context—i.e., the sequence /It/—can never occur in a morphology built upon morphemes drawn from the dictionary (Frantz and Russell, 1995), where the grapheme <I> is not used. Instead, the few putative /It/ sequences are represented as <ist>, in accordance with Frantz and Russell (1995), and a reformulation of the gemination rule (cf. Figure 5.5) causes the /s/ in these specific contexts to be elided when the /t/ becomes velar (see below). The neutralization rule, which transforms /I/ to /i/ is omitted for the same reason. Note that the ellipsis . . . in Figure 5.1 alludes to the lexically conditioned transformations to be discussed below.

```
define phonology [  
  ... .o.  
  coalescence .o.  
  semivowelLoss .o.  
  gemination .o.  
  sConnection .o.  
  yReduction .o.  
  breaking .o.  
  oReplacement .o.  
  ihLoss .o.  
  presibilation .o.  
  sssShortening .o.  
  semivowelDrop .o.  
  vowelShortening .o.  
  tAffrication .o.  
  postsibilation .o.  
  iAbsorption .o.  
  desyllabification .o.  
  glottalMetathesis .o.  
  vowelEpenthesis .o.  
  glottalReduction .o.  
  glottalLoss .o.  
  glottalAssimilation .o.  
  accentSpread .o.  
  breakDelete .o.  
  iLoss  
] ;
```

Figure 5.1: Parser 1 phonology: ordered phonological rules.

The 24 figures from Figure 5.4 to Figure 5.26 provide the formulations of the rules-as-FSTs listed in Figure 5.1. Aspects of individual rules are discussed when they are relevant either in terms of explaining how to achieve certain effects using the foma rewrite rule syntax or when they deviate significantly from the formulations in Frantz (1991, pp. 152–155). Some of the

rules make reference to regular expressions that are previously defined in the script and which encode sets of phonemes; these are provided in Figure 5.2. Also, the square brackets enclosing FST definitions are simply a device that allows the definitions to be broken across several lines and therefore rendered more readable. Finally note that some of the rules (e.g., breaking Figure 5.8) perform transformations on the hyphen delimiter itself instead of on the empty string with the delimiter as part of the context. However, this is not a significant fact since the rules could all be rewritten to leave the delimiters intact and have subsequent rules ignore them as necessary; the formulations presented here are justified by their ability to pass the tests that the phonology is designed to account for and their transparency of expression.

```
define phonemes [ p | t | k | m | n | s | w | y | h | "" |  
  a | i | o | á | í | ó ] ;  
define vowels [ a | i | o | á | í | ó ] ;  
define accentedVowels [ á | í | ó ] ;  
define consonants [ p | t | k | m | n | s | w | y ] ;  
define obstruents [ p | t | k | m | n | s ] ;  
define stops [ p | t | k | m | n ] ;  
define plosives [ p | t | k ] ;  
define glides [ w | y ] ;
```

Figure 5.2: Phoneme class regular expressions.

The foma script formulation of coalescence in Figure 5.3 demonstrates use of the comma to separate two distinct transformations that occur in a single environment (cf. Hulden, 2012). It also illustrates how enclosing

a regular expression in parentheses indicates optionality; i.e., $w \text{ "-" } i \text{ (i)}$ matches $/w-i/$ or $/w-ii/$. This rule is given as $wi(:) \rightarrow o$ in Frantz (1991, p. 152).

```
define coalescence [
  w "-" i (i) -> o ,
  w "-" i (i) -> ó ||
  _ [ consonants | h | "" ] ] ;
```

Figure 5.3: Coalescence.

The semivowel loss rule of Figure 5.4 is given in Frantz (1991, p. 154) as $G \rightarrow \emptyset / C_ ,$ where $C \neq ' .$

```
define semivowelLoss glides -> 0 || obstruents "-" _ ;
```

Figure 5.4: Semivowel loss.

The current phonology's implementation of the gemination rule (Figure 5.5) of Frantz (1991, p. 152) is markedly different from the source formulation. In the grammar, it is stated as a general rule of the form $C_1 \rightarrow C_2/_+C_2$. However, this generality is not supported by the data. In fact, the epenthesis of $/I/$ and $/oh/$ between adjacent plosives across morpheme boundaries—what Frantz (1991, pp. 78–79) classifies as morpheme-initial variation—will bleed gemination in the vast majority of cases where it would otherwise occur. The gemination rule actually appears to be specifically de-

signed for a very particular set of environments where the plosives are not actually underlyingly adjacent across a morpheme boundary but are put into that configuration as a result of the deletion of the leftmost vowel of the inverse suffixes /-oki/, /-okoo/, and /-ok/ when these follow /t/ (cf. the inverse clipping rule of Figure 5.45 below). Since this gemination transformation bleeds s-insertion, and since s-insertion (as explained above) is not used in the present phonology, the gemination rule as formulated here is restricted to transforming /t-k/ to /k-k/ (e.g., to produce [ohkokk] from /ohkot-k/) and to transforming /anist-k/ to /anikk/ in order to capture the peculiar behaviour of /waanist/ ‘say’ and its cognates. In short, whereas Frantz (1991) posits inverse suffix allomorphy, gemination, and then neutralization to produce the transformation sequence /anIt-ok/ → /anIt-k/ → /anIk-k/ → /anikk/, I posit inverse suffix allomorphy followed by a lexically constrained gemination which elides /s/ and produces the transformation sequence /anist-ok/ → /anist-k/ → /anikk/.

```
define gemination [
  [ a n i s t -> a n i k || _ "-" k ] .o.
  [ t -> k || _ "-" k ] ] ;
```

Figure 5.5: Gemination.

The s-connection A and B rules of Frantz (1991, p. 152) are $\emptyset \rightarrow s/C+_s$ and $\emptyset \rightarrow i/V(‘)+_s$, respectively. In the phonology script of Parser 1, s-connection is formulated as a single rule (cf. Figure 5.6). According to Frantz

(1991), the s-connection B rule does not apply when the /s/ of the context is part of a suffix. However, since suffixes cannot be readily identified in simple concatenations of strings of morphemes and delimiters,²⁴⁵ the present phonology captures the variability by making this transformation optional. This is accomplished by parenthesizing the arrow digraph, i.e., via (->). Although this strategy results in overgeneration, it will still produce the correct underlying representation during analysis; any incorrect analyses produced as a result of this optionality should be filtered out by the morphology FST. Another way of saying this is that this phonology will generate for /á-sínaaki-wa/ both the correct surface realization <áísínaakiwa> and the incorrect one <ásínaakiwa>. Any morphologically possible phoneme sequence that does generate the correct surface form but which is not the correct analysis, should be ranked as improbable by the *N*-gram LM disambiguator.

A technical device illustrated by the formulation of the s-connection rule is the use of [.] in Figure 5.6 to represent the empty string. In general, [.] should be used to represent the empty string on the left-hand side of a rewrite rule while 0 should be used on the right-hand side (cf. Figure 5.4).

²⁴⁵With a *rich upper* morphology it would, however, be possible to identify Blackfoot suffixes. That is, one could constrain a rule such as s-connection to occur only in environments with a stem somewhere to the left, where *stem* is identified via some regular expression over categories.

```
define sConnection [
  [ "-" -> s || stops _ s ] .o.
  [ [..] (->) i || vowels ("") "-" _ s ] ] ;
```

Figure 5.6: s-connection.

Figure 5.7 shows Parser 1's implementation of the y-reduction rule of Frantz (1991, p. 154), which is given in the source as $iyi \rightarrow ii/C_y$. Frantz (1991) notes that the <y> is sometimes pronounced in careful speech and, as a result, may be transcribed in the orthography. This is the reason for the optionality in the current formulation of the y-reduction rule.

```
define yReduction [ y (->) 0 ||
  [ obstruents | "" ] [ i | í ] _ [ i | í ] ("") y ] ;
```

Figure 5.7: y-reduction.

Parser 1's phonology's implementation of the breaking rule of Frantz (1991, p. 152) is given in Figure 5.8 and deviates from the formulation there, viz. $k \rightarrow ks/_I$. If breaking /I/ were used in the phonemico-orthographic transcriptions of the dictionary-derived lexicon, then $"-" \rightarrow s \ || \ k \ _ \ I$ would be an appropriate formulation. However, since /I/ is not used, the strategy taken here is again to employ optionality. This optionality and the resulting overgeneration could be avoided if lexical conditions on the environment were specified or, perhaps better, if some combination of lexical

5.2. Parser 1

and morphological domain conditions were specified (cf. Armoskaite, 2006). However, for reasons of expediency I leave this to future development of the parser.

```
define breaking "-" (->) s || k _ [ i | í ] ;
```

Figure 5.8: Breaking.

Figure 5.9 shows Parser 1's implementation of the o-replacement rule of Frantz (1991, p. 152), given there as $o \rightarrow a/_+a$, where $+a$ is not a suffix. Like s-connection B above, this rule also requires that the right-hand side of the context not be part of a suffix. Again, the approach taken here is therefore to make the transformation optional. In addition, the rule as implemented here allows for cross-speaker variation whereby $\langle o \rangle$ is either deleted or replaced by $\langle a \rangle$ (cf. Frantz, 1991, p. 152, fn. 165). This is effected by the disjunction in the regular expressions on the right-hand sides of the arrows.

```
define oReplacement [
  [ o (->) [ a | 0 ] , ó (->) [ á | 0 ] || _ "-" a ] .o.
  [ [ o | ó ] (->) [ á | 0 ] || _ "-" á ] ] ;
```

Figure 5.9: o-replacement.

The ih-loss rule, which is given in Frantz (1991, p. 153) as $ih \rightarrow \emptyset/s_s$, is formulated here as in Figure 5.10.

```
define ihLoss [ i | í ] "-" h -> 0 || s _ s ;
```

Figure 5.10: ih-loss.

Figure 5.11 shows Parser 1's implementation of the presibilation rule of Frantz (1991, p. 153), given in the source as $ihs \rightarrow ss$. While Frantz (1991, fn. 167) points out speaker- and dialect-level variation in this rule, this is not captured by the current implementation.

```
define presibilation [ i | í ] "-" h -> s || _ s ;
```

Figure 5.11: Presibilation.

The sss-shortening rule is formulated as $sss \rightarrow ss/_C$ in Frantz (1991, p. 155) and is formulated in the present phonology as in Figure 5.12.

```
define sssShortening s -> 0 || _ s s [ stops | glides ] ;
```

Figure 5.12: sss-shortening.

The semivowel drop rule, which is given as $G \rightarrow \emptyset/\#_$ in Frantz (1991, p. 153), is formulated here as in Figure 5.13.²⁴⁶

²⁴⁶The $\.#.$ sequence in Figure 5.13 and elsewhere denotes the left or right edge of a string.

```
define semivowelDrop glides -> 0 || .#. _ ;
```

Figure 5.13: Semivowel drop.

The vowel shortening rule is formulated as $V_i: \rightarrow V_i/_+V$ in Frantz (1991, p. 153) and is formulated in the present phonology as in Figure 5.14.

```
define vowelShortening [
  [ [ a | á ] -> 0 || [ a | á ] _ "-" vowels ] .o.
  [ [ i | í ] -> 0 || [ i | í ] _ "-" vowels ] .o.
  [ [ o | ó ] -> 0 || [ o | ó ] _ "-" vowels ] ] ;
```

Figure 5.14: Vowel shortening.

The t-affrication rule of Frantz (1991, p. 154) is given as $t \rightarrow ts/_i$ in the source and is formulated in the present phonology as in Figure 5.15.

```
define tAffrication "-" -> s || t _ [ i | í ] ;
```

Figure 5.15: t-affrication.

The postsibilant rule of Frantz (1991, p. 154) is given in the source as $ih \rightarrow s/s_$ and is formulated here as in Figure 5.16.

```
define postsibilation [ i | í ] ("") h -> s || s _ ;
```

Figure 5.16: Postsibilation.

Figure 5.17 shows Parser 1's implementation of the i-absorption rule of Frantz (1991, p. 153), which is formulated in the source as $i \rightarrow \emptyset / s_{-}\{a,o\}$. Note that this rule may, in fact, be more complicated since there is evidence that the lexical prominence on the deleted /i/ may persist on the first vowel of the right-hand side of the context, cf. <o'tsóawaistsi> 'their hands' (Frantz, 1991, p. 69) which is the surface realization of /w-mo'tsís-oaawaistsi/ after i-absorption has applied.²⁴⁷ Further work is needed to determine whether this is truly the result of lexical prominence persisting after elision of its host vowel.

```
define iAbsorption [ i | í ] ("") -> 0 || s _ [ a | á | o | ó ] ;
```

Figure 5.17: i-absorption.

The desyllabification rule of Frantz (1991, p. 153) is formulated in the source as $\{ i \rightarrow y, o \rightarrow w \} / V+_V$ and in the present phonology as in Figure 5.18.

²⁴⁷Note that the /s/ at the end of /mo'tsís/ 'hand' is a so-called "non-permanent" consonant (Frantz, 1991, p. 9), which accounts for its elision before a vowel-initial suffix (see Figure 5.40 below).

```

define desyllabification [
  [ "-" i -> y || vowels _ [ a | á | o | ó ] ] .o.
  [ "-" o -> w || vowels _ [ a | á | i | í | ó ] ] ] ;

```

Figure 5.18: Desyllabification.

Figure 5.19 shows Parser 1's implementation of the glottal metathesis rule of Frantz (1991, p. 154), which is given in the source as 'V → V'/V_C. The twelve comma-separated transformations are required in order to effect metathesis with each possible vowel type.²⁴⁸ As the phonology is developed and tested, it may become necessary to create additional transformations for the /ao/, /ai/, and /oi/ sequences and for rising or falling prominence contours on long vowels, e.g., /aá/.

²⁴⁸Note that it is not possible to make use of indexed backreferences in foma script rewrite rules. It is possible to use `. . .` to refer back to the *entire* left-hand side of a transformation in the right-hand side. Thus `[a | b] -> X . . . X ;` will map *cat* to *cXaXt* and *bit* to *XbXit*. However, one cannot refer back to select portions of the sequence matched by the left-hand side regular expression as is possible in the regular expression interfaces of modern programming languages. If this were possible, then the glottal metathesis could be written in a much more general and succinct way.

```

define glottalMetathesis [
  "" "-" a -> "-" a "" ,
  "" "-" á -> "-" á "" ,
  "" "-" a a -> "-" a a "" ,
  "" "-" á á -> "-" á á "" ,
  "" "-" i -> "-" i "" ,
  "" "-" í -> "-" í "" ,
  "" "-" i i -> "-" i i "" ,
  "" "-" í í -> "-" í í "" ,
  "" "-" o -> "-" o "" ,
  "" "-" ó -> "-" ó "" ,
  "" "-" o o -> "-" o o "" ,
  "" "-" ó ó -> "-" ó ó "" ||
  vowels _ [ consonants | "" | h ] ] ;

```

Figure 5.19: Glottal metathesis.

Parser 1's implementation of the vowel epenthesis rule of Frantz (1991, p. 155) is given in Figure 5.20. The source formulation is $\emptyset \rightarrow V_i/V_i'_h$. The FST formulation in Figure 5.20 allows for the speaker-level variation wherein some epenthesize a vowel and others elide the glottal stop (cf. Frantz, 1991, fn. 169).

```

define vowelEpenthesis [
  a "" -> [ a "" a | a ] ,
  á "" -> [ á "" á | á ] ,
  i "" -> [ i "" i | i ] ,
  í "" -> [ í "" í | í ] ,
  o "" -> [ o "" o | o ] ,
  ó "" -> [ ó "" ó | ó ] || _ h ] ;

```

Figure 5.20: Vowel epenthesis.

The glottal reduction rule is formulated here as in Figure 5.21 and is given in the source (Frantz, 1991, p. 154) as $' \rightarrow \emptyset / _'$.

```

define glottalReduction "" "" -> "";

```

Figure 5.21: Glottal reduction.

Frantz (1991, p. 154) points out that glottal loss ($' \rightarrow \emptyset / \text{VV} : _ \text{C}$) does not occur when the vowel before the glottal is a variable-length vowel. However, Figure 5.22 does not capture this restriction. Without modifying the lexicon to indicate variable-length vowels, encoding this restriction in the phonology would entail redundantly specifying the same lengthy lexical contexts in both the glottal loss rule and in the variable-length vowel shortening transformation discussed below. I leave this as an unsolved problem.

```

define glottalLoss [
  a a "" -> a a ,
  á á "" -> á á ,
  i i "" -> i i ,
  í í "" -> í í ,
  o o "" -> o o ,
  ó ó "" -> ó ó || vowels ("-") _ consonants ] ;

```

Figure 5.22: Glottal loss.

The glottal assimilation rule, which is given in Frantz (1991, p. 154) as $V_i' \rightarrow V_i:/_ (s)C:$, where $C \neq s$, is formulated in the present phonology as in Figure 5.23.

```

define glottalAssimilation [
  a "" -> a a || ,
  á "" -> á á || ,
  i "" -> i i || ,
  í "" -> í í || ,
  o "" -> o o || ,
  ó "" -> ó ó || _ (s) [ p p | t t | k k | m m | n n ] ] ;

```

Figure 5.23: Glottal assimilation.

The accent spread rule is given as $V \rightarrow \acute{V}/\acute{V}+_$ in Frantz (1991, p. 155)²⁴⁹ and is formulated here as in Figure 5.24.

²⁴⁹Frantz (1991) actually uses “[+accent]” in the formulation of the accent spread rule. However, the reformulation given here expresses the same transformation and is easier to write.

```
define accentSpread [  
  a -> á ,  
  i -> í ,  
  o -> ó || accentedVowels "-" _ ] ;
```

Figure 5.24: Accent spread.

As mentioned above, the break delete rule (Figure 5.25) elides the morpheme delimiter. While a much simpler formulation would be `"-" -> 0`, the one used is superior insofar as it reduces ambiguity during parsing since it assumes that the underlying representations cannot contain sequences of adjacent delimiters. This is accomplished by the use of `\"` which is the term complement (Beesley and Karttunen, 2003, ch. 2) of the hyphen character, i.e., the set of all single-character strings except the hyphen.

```
define breakDelete "-" -> 0 || [ .#. | \" ] _ \"
```

Figure 5.25: Break delete.

The final general phonological rule is i-loss (cf. Figure 5.26), which is given in Frantz (1991, p. 153) as $i \rightarrow \emptyset / Vy_{\{a,o\}}$.

```
define iLoss [  
  [ [ i | í ] -> 0 ||  
    [ a | á | o | ó ] y _ [ a | á | o | ó ] ] .o.  
  [ i y [ i | í ] -> i (i) y ,  
    í y [ i | í ] -> í (í) y || _ [ a | á | o | ó ] ] ] ;
```

Figure 5.26: i-loss.

Figure 5.27 shows the ordered lexical phonological rules that constitute the initial portion of the `phonology` FST whose latter portion was introduced in Figure 5.1. These lexically conditioned phonological rules are, for the most part, based on the patterns of allomorphy described in Frantz (1991, pp. 78–83).

```
define phonology [  
  irregularBreaking .o.  
  semivowelAlternation .o.  
  nasalInitialVerbs .o.  
  imperativeInitialVowelElision .o.  
  codafication .o.  
  stopStopEpenthesis .o.  
  become0 .o.  
  becomeA .o.  
  nasalLoss .o.  
  variableLengthVowels .o.  
  3mm .o.  
  DTPVowelElision .o.  
  nonPermanentConsonants .o.  
  diphthongization .o.  
  initialChangeAY .o.  
  initialChangeII .o.  
  yiLoss .o.  
  inverseClipping .o.  
  ...  
] ;
```

Figure 5.27: Parser 1 phonology: lexical phonological rules.

Certain verb stems cause the right-edge /t/ and /k/ of preceding prefixes to assibilate even though these stems do not ostensibly begin with a high front vowel. These are sometimes represented as beginning with breaking /I/, e.g., compare /Ioyi/ ‘eat’ (Frantz, 1991, p. 85) to /ooyi/ ‘eat’ (Frantz and Russell, 1995, p. 170). Figure 5.28 shows the irregular breaking rule which effects this alternation by converting the initial segment of these verb stems to /i/ in the relevant environments. There are probably other verb stems that behave in this way; the parser can be used as a tool to discover them.

```

define irregularBreaking [
  [ o -> i || [ t | k ] "-"
    _ o [ y i | w a t ( o o ) ] [ .#. | "-" ] ] .o.
  [ y -> i || [ t | k ] "-"
    _ [ o ' k a a | á a p i ] [ .#. | "-" ] ] ] ;

```

Figure 5.28: Irregular breaking.

The semivowel alternation rule of Figure 5.29 handles the epenthesis of a semivowel after a certain set of morphemes when these precede a vowel-initial morpheme. The epenthesized semivowel is /y/ before high front vowels and /w/ before other vowels. These morphemes may be analyzed as being terminated by a semivowel that is underspecified for place of articulation Frantz (1991, pp. 82–83). In Frantz and Russell (1995) they are indicated by a period at the end of the morpheme.²⁵⁰

²⁵⁰However, these periods were removed when the Blackfoot data set described in § 4.2 was created.

```

define SVAItMorphs [
  i h t a |
  i h t s o o h k i t s i |
  i k a |
  i k k i a "" |
  i k s i s s t o |
  i n s s t a |
  i s a k o o |
  i t s i |
  m a t s i |
  n a a t o |
  o h p i i |
  o h t o |
  s o p o |
  w a a h k o |
  w a a k o o "" |
  w a a w o ] ;
define semivowelAlternation [
  [ [..] -> y ||
    [ .#. | "-" ] SVAItMorphs _ "-" [ i | í ] ] .o.
  [ [..] -> w ||
    [ .#. | "-" ] SVAItMorphs _ "-" [ o | ó | a | á ] ] ] ;

```

Figure 5.29: Semivowel alternation.

Some verb stems which appear to have vowels at their left edges, actually surface with a nasal at the left edge when prefixes are absent (cf. Frantz, 1991, p. 80). This phenomenon is the verbal counterpart of nominal nasal loss (Figure 5.36) and is to some degree an artifact of the different conventions for the transcription of verbal and nominal forms in (Frantz and Russell, 1995). The nasal-initial verbs rule in Figure 5.30 implements this alternation

for a small set of verb stems, e.g., /yaamaahkiaaki-t/ → <namaahkiaakit> ‘sweep!’ (Frantz and Russell, 1995). Here the environment where a nasal is introduced is specified as word-initial position in imperative or nominalization constructions.²⁵¹ Note that perusal of the data in Frantz and Russell (1995) indicates that there may be many more verb stems (not included in Figure 5.30) that begin with a nasal when in word-initial position. I leave the incorporation of these to future work on the parser. Finally, note the definition and use of the imperative/nominalization suffix regular expression `impNomSfx` which matches the shape of the relevant suffixes.²⁵²

²⁵¹It would be interesting to discover whether this word-initial nasal insertion also occurs for these particular verb stems in other prefix-less environments, e.g., in the subjunctive mood. If so, this would suggest that the context is more general, i.e., that this happens word-initially. If this is the case, then it would constitute evidence that the so-called initial change alternations that correlate with past tense interpretation, e.g., *inikíwa* ‘he sang’, and where no nasal is introduced do actually involve prefixation of some morpheme like /ii-/.

²⁵²This context should probably be generalized to include the imperative suffixes of the transitive animate paradigm, cf. Frantz (1991, p. 151).

```

define impNomSfx "-" [ t | n | h s i n ] [ "-" | .# . ] ;
define nasalInitialVerbs [
  [ y a (a) -> n a || .# . _
    m a a [ h k i a a k i | k i o k o "" s i ]
    impNomSfx ] .o.
  [ y -> n      || .# . _
    á á p i i "" p o y i impNomSfx ] .o.
  [ [..] (->) n  || .# . _
    [ i n i h k i | i p a i t a p i i y i ] impNomSfx ] .o.
  [ [..] -> n   || .# . _
    i i t s i "" p o y i impNomSfx ] .o.
  [ [..] -> n i  || .# . _
    i p [ ó í s o y a a w a n i | o w á ó o ]
    impNomSfx ] .o.
  [ i s -> n a a  || .# . _
    t s i k a p ó ó y i n n i impNomSfx ] .o.
  [ i s -> n a a  || .# . _
    t s i k a p o i s í n a i i impNomSfx ] .o.
  [ o -> m a      || .# . _
    t o o m [ phonemes | "-" ]+ impNomSfx ] ] ;

```

Figure 5.30: Nasal-initial verbs.

Figure 5.31 shows a rule which deletes word-initial /oh-/ and /i-/ sequences in imperative constructions (cf. Frantz, 1991, pp. 78–79).²⁵³ It also transforms the locative prefix /it-/ to /ist-/ in the same environments (cf. Frantz, 1991, p. 96). Note that the complex context in the second of the three composed FSTs in Figure 5.31 (lines 5–6) is needed in order to prevent /ist-/ from becoming /st-/. Also, the `consonants` regular expression here prevents /#ihp/ from becoming /#hp/.

²⁵³It may be that this rule is more general and that these sequences disappear in nominalizations also.

```

define imperativeInitialVowelElision [
  [ [..] -> s || .#. (i) i _
    t "-" [ phonemes | "-" ]+ "-" t .#. ] .o.
  [ i -> 0 || .#. _
    [ consonants - s ] [ [ phonemes | "-" ] - s ]
    phonemes [ phonemes | "-" ]* "-" t .#. ] .o.
  [ o h -> 0 || .#. _
    consonants [ phonemes | "-" ]+ "-" t .#. ] ] ;

```

Figure 5.31: Initial vowel elision in imperatives.

The codafication rule of Figure 5.32 handles the idiosyncratic behaviour of certain C_1VC_2 -initial stems (e.g., /ponoká/ ‘elk’) which do not trigger epenthesis after stop-final prefixes (cf. Figure 5.33) but instead have their leftmost C_1V sequence elided and replaced with iC_2 .²⁵⁴ That is /sikponoká-wa/ surfaces as <siksinnokawa> and not as *<sikohponokawa> or *<siksiponokawa> (cf. Frantz, 1991, p. 79).

```

define codafication [
  p o n -> i n n || stops "-" _ o k [ "-" | .#. ] .o.
  k i i p -> i p p || stops "-" _ [ "-" | .#. ] .o.
  k i p -> i p p || stops "-" _ i t a [ "-" | .#. ] ] ;

```

Figure 5.32: Codafication.

The stop-stop epenthesis rule (Figure 5.33) inserts /i-/ or /oh-/ when plosives meet at a morpheme boundary. It can be seen as the counterpart

²⁵⁴The “codafication” label arises from viewing this transformation as C_1 moving to coda position and taking on the place of articulation of C_2 .

of the initial vowel elision alternation presented in Figure 5.31. In fact, both of these rules are really artifacts of the way in which the dictionary (Frantz and Russell, 1995) data are transcribed, such that verb stems are recorded with word-initial /i/ and /oh/ if they have allomorphs of that nature while noun stems are not (cf. Frantz, 1991, pp. 78–79). Note that /náámayi/ ‘bow’ seems to be anomalous in being the only nasal-initial stem that participates in this alternation; in most other cases, the nasal is elided by the nasal loss rule (Figure 5.36) (cf. Frantz, 1991, p. 80). Note in addition that this stop-stop epenthesis rule overgeneralizes since it allows for the insertion of either /i/ or /oh/ even though most noun stems seem to do one or the other (with the exception of /piitaa/ ‘eagle’, as Frantz (1991, fn. 94) points out).

```
define stopStopEpenthesis [
  [..] -> [ i | o h ] ||
  plosives "-" _ [ plosives | n á á m a y i ] ] ;
```

Figure 5.33: Stop-stop epenthesis.

A few noun stems that are listed in the lexicon as beginning in /i/ or /ma/ look as though they begin in /o/ when they are prefixed by morphemes ending in a plosive (Frantz, 1991, p. 80). Similarly, some verb stems that are listed as beginning in /o/ change this vowel to /a/ in the same word-initial imperative/nominalization environment referenced in the nasal-initial verbs rule (Figure 5.30). These two alternations, which are captured by the become /o/ (Figure 5.34) and become /a/ (Figure 5.35) rules, respectively,

probably apply to more stems than are captured here.

```
define become0 [
  [ i -> o  || plosives "-" _ m i t á á ] .o.
  [ m a -> o || plosives "-" _ [ m í í | n i s t s í ] ] .o.
  [ i s (->) o || "-" _ t t o á n ] ] ;
```

Figure 5.34: Become /o/.

```
define becomeA [
  o (->) a || .#. _ k [ a a | s t a k i ] impNomSfx ] ;
```

Figure 5.35: Become /a/.

```
define nasalLoss [ m | n ] -> 0 || consonants "-" _ ;
```

Figure 5.36: Nasal loss.

Certain vowels transcribed as long are actually variable in length. These vowels are long in a particular set of environments, are variably long or short in one specific environment, and are short elsewhere (Frantz, 1991, pp. 80–81). They are occasionally transcribed in Frantz (1991) using the convention <V:>, e.g., <wa:nii>; however, they are not indicated in Frantz and Russell (1995). As a result, there are probably many more variable-length vowels in the lexicon than are captured by the FST specified in Figure 5.37. Instead of composing complex regular expressions to capture the relevant environments, the implementation of the variable-length vowels rule simply

optionally shortens these vowels in all environments. Note that the vowel-initial theme suffixes (i.e., direct and inverse) are all variable-length vowels; an inconsistency in the data set is that the variable-length vowels of the inverse suffixes /-ok/, /-okoo/, and /-oki/ are transcribed as short. The rule in Figure 5.37 handles these details.

```

define variableLengthVowels [
  [ a a (->) a | |
    [ .#. | "-" ] w _
      [ n i i | n i s t | n i s t o o |
        w a h k a a |
        w a y á k i |
        y á k i |
        s a i "" n i |
        h k a y i |
        w a k a a |
        m i ] [ .#. | "-" ] ,
    "-" _ "-" ,
    [ .#. | "-" ] s _ k s i [ .#. | "-" ] ,
    [ .#. | "-" ] _ [ n i s t | k i ] [ .#. | "-" ] ] .o.
  [ o o (->) o | | "-" _ "-" ] .o.
  [ o (->) o o | | "-" _ k ( [ o o | i ] ) "-" ] ] ;

```

Figure 5.37: Variable-length vowels.

In particular environments, Certain intransitive verbs are suffixed by /mm/. According to Frantz (1991, p. 82), the /mm/ appears when semivowel-initial suffixes immediately follow the verb stem. According to Frantz and Russell (1995, p. xvii) the /mm/ appears with third person subjects in independent clauses. The rule in Figure 5.38 inserts /mm/ between any of these verb stems and any of the third or obviative third person singular or plural

suffixes, i.e., /-wa/, /-yini/, and /-yi/. The relevant verbs are identified in Frantz and Russell (1995) by the presence of the string *3mm*;²⁵⁵ the implementation here lists these stems in the environment. Note that /ikaissi/ ‘be old’ has optional /-mm/-suffixation (Frantz and Russell, 1995, p. 35).

```

define 3mm [
  [ [..] -> m m || [ .#. | "-" ] [
    i k o ' p o |
    i k s i s á í i k i |
    i k s i s t o ' s i |
    i m i |
    i n á á m m |
    i n n o ' s i |
    i p a h k s i m i |
    p á p i s a |
    i p a p o |
    i p i i |
    ... ]
  - "-" [ w a | y i | y i n i ] [ "-" | .#. ] ] .o.
  [ [..] (->) m m ||
    [ .#. | "-" ] i k a i s s i
    - "-" [ w a | y i | y i n i ] [ "-" | .#. ] ] ] ;

```

Figure 5.38: 3mm.

The distinct third person verbal (DTP) suffixes /-áyi/ (singular), /-aiksi/ (animate plural), and /-aistsi/ (inanimate plural) occur after the agreement suffixes when there is another third person in the discourse context (Frantz, 1991, p. 48). Any vowel that immediately precedes one of these suffixes is

²⁵⁵For brevity, Figure 5.38 does not list all of the *3mm* verbs in the actual FST definition. The ellipsis ... represents an additional 34 *3mm* verbs.

elided, as implemented in the DTP vowel elision rule of Figure 5.39.

```
define DTPVowelElision [ vowels -> 0 ||  
  _ "-" [ á y i | a i [ s t s i | k s i ] ] [ .#. | "-" ] ] ;
```

Figure 5.39: Vowel elision before the DTP suffixes.

Certain stem-final /n/, /m/, and /s/ consonants are identified as non-permanent (Frantz, 1991, pp. 9, 81–82), meaning that they disappear before vowel-initial suffixes. While the grammar (Frantz, 1991) occasionally uses capitalized graphemes to indicate these consonants, the dictionary (Frantz and Russell, 1995) does not make use of this convention. The implementation in Figure 5.40 optionally elides morpheme-final /n/ and /s/ before vowel-initial suffixes and regularly elides the /m/ that terminates a small set of morphemes identified in Frantz (1991); this special treatment of /m/ accords with the discussion in the grammar which suggests that a large number of morpheme-final /n/ and /s/ are non-permanent but only a few morpheme-final /m/ are.

```

define nonPermanentConsonants [
  [ [ n | s ] (->) 0 || _ "-" vowels ] .o.
  [ m -> 0 || [
    á t o "" a h s i |
    m á í i p s s i |
    a s ó k a "" s i ] _
    "-" vowels ] ] ;

```

Figure 5.40: Non-permanent consonants.

The first person inclusive plural agreement suffixes /-o'p(a)/, /-o's(i)/, and /-o'k(i)/ cause the word-final vowels of certain morphemes to become /ɔ/ (i.e., orthographic <ao>), thereby producing the diphthong [ɔo] (cf. Frantz, 1991, p. 83). The diphthongization rule in Figure 5.41 effects this lexically conditioned transformation. Note the parenthesization of the <o> in the right-hand sides of the replacements; this is done because the surface form is sometimes transcribed as <ao>, cf. <iipiksao'pa> 'we hit' (Frantz, 1991, p. 83).²⁵⁶

²⁵⁶If the orthographic transcription <iipiksao'pa> is truly accurate, i.e., if it really represents a phonetic transcription [i:piksɔ'pə], then "diphthongization" is something of a misnomer.

```

define 21Sfxs "-" o "" [ p (a) | [ s | k ] (i) ] [ "-" | .#. ] ;
define diphthongization [
  [ o o -> a (o) || [ .#. | "-" ] [
    i h t s i s |
    k i "" t s i i k s o w |
    i s t t s s |
    i t a p |
    o h t s s a p |
    o p a s |
    s i s t s i k |
    s s k |
    w a a w a a t t s i s o w |
    y i i s t a p ] _ 21Sfxs ,
  [ .#. | "-" ] _ 21Sfxs ] .o.
[ ó o -> a (o) || [ .#. | "-" ] s o w _ 21Sfxs ] .o.
[ i i -> a (o) || [ o | ó ] p _ 21Sfxs ] .o.
[ i -> a (o) || [ .#. | "-" ]
  [ i s t a w a "" s | i p i k s ] _ 21Sfxs ] ] ;

```

Figure 5.41: Diphthongization.

Frantz (1991, p. 36) points out that a small number of verb stems that begin in /sV/ or /iCV/ replace this initial sequence with /sayV/ or /CayV/, respectively, when there are no prefixes present. These verb stems can be identified in the dictionary (Frantz and Russell, 1995) by the presence of the annotation *init chg*. The initial change /-ay-/ rule in Figure 5.42 implements this alternation. Based on the data in the dictionary, this alternation is optional for some stems, hence the instances of (->) in the rule formulation.

```

define initialChangeAY [
  [ i k -> k a y || .#. _ [
    i i |
    i p p |
    i "" t a w a a t o o |
    i "" t a y i s s k s i m m a a ] [ .#. | "-" ] ] .o.
  [ i k s -> k a y ||
    .#. _ i s s t a "" p s s i [ .#. | "-" ] ] .o.
  [ i p -> p a y ||
    .#. _ i p o "" t a m i a p i k s s a t t s i i y i
    [ .#. | "-" ] ] .o.
  [ i p (->) p a y || .#. _ i p o t t a a [ .#. | "-" ] ] .o.
  [ k s (->) k a y || .#. _ i k k [ .#. | "-" ] ] .o.
  [ s -> s a y ||
    .#. _ [ a "" k a p | a t t s i k s s k a a ]
    [ .#. | "-" ] ] .o.
  [ s (->) s a y || .#. _ o o t a a [ .#. | "-" ] ] .o.
  [ i k s -> k s y i ||
    .#. _ k s k a a a t [ .#. | "-" ] ] .o.
  [ i k a -> k || .#. _ y i h t s i [ .#. | "-" ] ] .o.
  [ i k a -> [ k a a | i i k ] ||
    .#. _ y i n n i [ .#. | "-" ] ] ] ;

```

Figure 5.42: Initial change /-ay-/.

The second type of initial change involves the prefixation of /ii-/ to prefix-less verb stems or the changing of the initial vowels of such verb stems to /ii-/ (Frantz, 1991, p. 36). The rule implementing such /ii-/ initial change in Figure 5.43 exploits the fact that (given the conventions of the dictionary) almost all verbs begin with a vowel, a semivowel, or <s>²⁵⁷ in

²⁵⁷In the current data set (the lexicon of which is, as discussed, almost entirely from (Frantz and Russell, 1995)), 97% of the verb stems (2,517 of 2,596) begin with a vowel, a semivowel, or <s>. Also as discussed above, this is probably less a significant fact about

order to reduce the amount of overgeneralization (i.e., to non-verbs) that a rule of this type would otherwise produce.

```

define initialChangeII [
  [ [..] (->) i i ||
    [ .#. | .#. [ n i t | k i t | o t ] "-" ]
    _ [ s | w | y ] ] .o.
  [ vowels+ (->) i i ||
    [ .#. | .#. [ n i t | k i t | o t ] "-" ]
    _ [ consonants | "" | h ] ] ] ;

```

Figure 5.43: Initial change /-ii-/.

Certain verb stems lose a right-edge /yi/ when third person inflectional morphemes are suffixed (Frantz and Russell, 1995, p. xvii). The dictionary indicates these verb stems via *yi Loss*.²⁵⁸ The rule in Figure 5.44 implements this alternation for the verb stems listed as having *yi Loss* in Frantz and Russell (1995).

Blackfoot and more an effect of the choice to transcribe verb stems as vowel-initial even in those cases where a consonant-initial allomorph is attested.

²⁵⁸I assume that /-yi/ loss only occurs after the third (and obviative third) person indicative suffixes /-wa/, /-yi/, and /-yini/. If the effect is seen in other moods, then this rule will need to be modified. Note also that sometimes the dictionary parenthesizes these /yi/ sequences, indicating that /-yi/ loss may be optional. Further work may be required to determine the nature of the variability implied by this.

```

define yiLoss [
  y i -> 0 || [ .#. | "-" ] [
    i k k a n i k s o o h p i "" |
    i n n i s i "" |
    i p a i t a p i i |
    i t á "" p a o h p i "" |
    i t a p i i |
    i "" t o m i k á ó h p a i "" p i |
    o h p i "" |
    o "" t s s o p o k i i |
    s a a p o h p i "" |
    s o y o o h p i "" |
    s s k s s p o h p i "" |
    w á á p a t s i i "" |
    w a a t a p o k a "" ]
  -
  [ y i | w a | y i n i ]
  [ .#. | "-" ] ;

```

Figure 5.44: /-yi/ loss.

Finally, the /o/ of the inverse theme suffixes /-ok/, /-okoo/, and /-oki/ disappears before verbs that end in /t/ (Frantz, 1991, pp. 57–58) (Figure 5.45).

```

define inverseClipping o -> 0 || t "-" _ k ( [ o o | i ] ) "-" ;

```

Figure 5.45: Inverse clipping.

This section has presented the phonology FST script of Parser 1 in some

detail. This phonology is based closely on the phonological and allomorphic analysis of the grammar (Frantz, 1991). Notable deviations from the generalizations of the grammar involve the use of optional transformations where complex contexts are either not fully known or are difficult to capture. The phonology presented here is a contribution to the linguistic analysis of Blackfoot since the rules and their orderings are fully explicit and computationally testable. Data points that cannot be accounted for by this phonology may reveal gaps in the analysis of the grammar or gaps in the interpretation and formalization of that analysis as formulated here. In either case, such data points may serve as useful starting points for deeper analyses of this component of the grammar of Blackfoot.

5.2.2 Morphology

The lexicon of the morphology of Parser 1 is extracted from the OLD corpus of morphemes which is defined explicitly in § 4.2.1. Generalizing somewhat, this morphemes corpus contains all of the lexically categorized forms, provided that each such form does not contain either a word delimiter or a morpheme delimiter (i.e., a space or a hyphen) in its morpheme break value (cf. the search expression (38) from § 4.2.1 above). This lexicon is essentially the set of headwords of the Blackfoot dictionary (Frantz and Russell, 1995) with some categorial modification and with a small number of additions coming from Blackfoot OLD contributors and the inflectional morphemes of Frantz (1991, pp. 147–151).

The morphotactic rules of the morphology of Parser 1 are simply the syntactic category string values of the corpus of morphologically well analyzed

word types which is described in § 4.2.2 above, cf. Table 4.8. There are, in total, 940 distinct morpheme category sequences that constitute well-formed words according to this morphology. Figure 5.46 provides a network diagram of a foma FST that encodes the 50 most common category sequences that are recognized by this morphology.²⁵⁹

²⁵⁹It would be interesting to compare these category string types ordered by counts to the templates for complex verbs and demonstratives that are provided by certain researchers (e.g., Bliss (2013)) in order to capture morphotactic generalizations. These category string types may help in discovering counterexamples to the templates and, thereby, in refining the generalizations.

tested on its training set and the success of Parser 1 as a whole rests (almost) entirely on the efficacy of the phonology and the LM.

As discussed in § 4.3.1 above, the categorization of agreement morphemes in the Blackfoot data set combined with the fact that the parser creator uses category sequences as morphotactic rules will result in an over-generative morphology. From a parsing perspective this is not a major issue since a morphology FST within a morphological parser need only recognize all valid morpheme sequences and filter out the bulk of the ungrammatical ones, so as to reduce the burden on the candidate ranker (LM) component.

However, it is also possible that the morphology described here could *under-generate*, if it were extracted from (i.e., trained on) and tested on disjoint sets, as is done for the LM ranker (cf. § 5.2.3). The likelihood of such under-generation is increased by the fact that the morphology-building algorithm of the parser creator does not result in a morphology that can recognize morphological analyses with unseen category sequences or with unseen morphemes (cf. § 3.1).

Future work may involve evaluation of the morphology-building algorithm of the parser creator. However, because this algorithm is so simple and because the odds seem stacked in favour of over-generation, I focus in this work on evaluating the phonologies and morpheme LM candidate

phemes are excluded from the morphemes corpus yet words containing these morphemes are not systematically excluded from the morphologically well analyzed words corpus, cf. § 4.2. Since morphemes categorized as *dem* are excluded, /amo/ *dem* is not in the lexicon and word analyses like /amo-ya/, /amo-iksi-ka/, and /amo/ will not be recognized; they should be analyzed as *dem-o* sequences, e.g., /am-o/ according to this morphology. The demonstrative suffix /hkayi/ ‘certain’ *agra* was also excluded from the lexicon because it is (for some reason) categorized as *agra* in the Blackfoot OLD, a categorization that is at odds with its distribution; this fact explains why the morphology of Parser 1 does not recognize word analyses like /ann-yi-hkayi/.

rankers of the Blackfoot parsers described here.

5.2.3 Language Model

The LM of Parser 1 is estimated from the set of 3,245 gold standard well analyzed word types, which is described in § 4.2.2 above. In order to reliably measure the success of the parser, it is necessary to ensure that the set of data used to evaluate it (i.e., the test set) is different from the set of data used to train its LMs. To do this, Parser 1 was created five times, each with a different LM that was trained on 2,920 (90%) randomly selected analyzed words of the gold standard analyzed word set. Each version of the parser was then tested against the remaining 325 (10%) words of the gold standard set that its LM was *not* trained on. The overall scores (F-scores and parse rates) given for the parser are averages of these five LM-specific versions.

The LM of Parser 1 is non-categorical, has order=3, and uses the modified Kneser-Ney (Hsu and Glass, 2008; Chen and Goodman, 1999) smoothing algorithm.

To get a sense of the size of these LMs, consider the following stats on one representative. This LM has 1,004 1-grams, 4,034 bigrams, and 5,917 trigrams. Table 5.1 shows the top 20 most frequent trigrams in one of Parser 1's LMs. The first column shows a readable shape-only representation of the trigram morpheme sequence, where # indicates a word edge. Column two shows the literal representation of the trigram in the LM and column three shows how many times it occurs in the training set (subset of the gold standard well analyzed words corpus).

The most frequent trigrams, as listed in Table 5.1, should not be sur-

5.2. Parser 1

trigram simple	trigram unambiguous	count
yi-aawa#	yi 3PL agrb aawa PRO oth </s>	88
ok-wa#	ok INV thm wa 3SG agrb </s>	63
yii-wa#	yii DIR thm wa 3SG agrb </s>	45
#nit-á	<s> nit 1 agra á IMPF asp	38
aa-wa#	aa DIR thm wa 3SG agrb </s>	36
#nit-áak	<s> nit 1 agra áak FUT ten	36
#nit-iik	<s> nit 1 agra iik INT adt	33
hp-wa#	hp DIR thm wa 3SG agrb </s>	26
inaa-mm#	inaa appear.as fin mm 3mm oth </s>	25
#it-á	<s> it LOC prev á IMPF asp	23
hp-yi#	hp NOM oth yi IN.SG num </s>	22
a`p-yi#	a`p be vrt yi be fin </s>	22
#nit-it	<s> nit 1 agra it LOC prev	22
#ii-ohpok	<s> ii PAST ten ohpok with adt	21
m-yii-wa	m TA fin yii DIR thm wa 3SG agrb	17
m-wa#	m DIR thm wa 3SG agrb </s>	17
mm-yi-aawa	mm 3mm oth yi 3PL agrb aawa PRO oth	17
m-yii#	m TA fin yii DIR thm </s>	16
inaa-mm-yi	inaa appear.as fin mm 3mm oth yi 3PL agrb	16
#nit-ino	<s> nit 1 agra ino see vta	16

Table 5.1: 20 most frequent trigrams in one of Parser 1’s training LMs.

prising to those familiar with Blackfoot morphology. By and large, they are sequences of two inflectional morphemes at word boundaries. It is interesting to note that several of the common trigrams contain morpheme representations of phenomena that are analyzed as morpho-lexically conditioned phonological alternations in the phonology of Parser 1. The reference here is to the \emptyset/mm (so-called *3mm*) alternation on certain intransitive verbs (compare rows 9, 17, and 19 in Table 5.1 to Figure 5.38 above) and the /ii-/ initial change phenomenon (compare the 14th row in Table 5.1 to Figure 5.43 above).

5.2.4 Results

The performance of Parser 1 is summarized in Table 5.2. The parse success rate for Parser 1—i.e., the percentage of parse attempts that result in the correct morphological analysis being generated—is 14%. It has an F-score of 0.32. Its precision is 0.53, indicating that 53% of the morphemes that it guesses are present in the relevant gold parse. Its recall is 0.23, indicating that it correctly guesses 23% of the correct morphemes in any given parse.

parse rate	F-score	precision	recall	phonology	morphon	LM
0.14	0.32	0.53	0.23	0.21	0.20	0.72

Table 5.2: Parser 1 results.

Form (43) is an example of a word that this parser correctly analyzes.

- (43) *itáísapsssiiststakio'piksi*
 it-á-sap-ssi-iststaki-o'p-iksi
 LOC-IMPF-in-wipe-back.forth-21PL-AN.PL
 prev-asp-adt-vrt-fin-agrb-num
 'washtubs' (Frantz and Russell, 1995, p. 29)

The phonology of Parser 1 generates the correct orthographic representation for a given morphological analysis 20.7% of the time. This accords with § 5.2.1 above where it was shown that this phonology can account for 21% of the mappings in the gold standard corpus.

The success rate of the morphophonology—i.e., the percentage of occasions where the correct analysis is among the set of candidates—is 19.5%.

Since, as discussed in § 5.2.2 above, the morphology recognizes 99.7% of the morpheme sequences in the gold standard corpus, it is expected that the success of the morphophonology will be just below that of the phonology.²⁶¹

The LM is successful 72% of the time. This means that when the morphophonology *does* generate the correct candidate in its set of parse candidates, then the LM is able to successfully find it 72% of the time.

5.2.5 Discussion

The performance metrics presented above indicate that the poor performance of the phonology is the primary impediment to the success of Parser 1. This section discusses two approaches to improving parser performance by modifying the phonology. The first involves removing vowel accent and segment length distinctions during generation; this approach is discussed at length because it is the one that is taken in the creation of Parser 2 (§ 5.3). The second approach, which is discussed briefly and left for future work, is to render the phonology more accurate by incorporating rules that generate non-lexical pitch accent marking, rules based on the work of Weber (2013), Weber and Allen (2012), and Kaneko (1999).

During evaluation, the mappings that the phonology was unable to account for were identified. Inspection of these reveals that 27.6% of them could be accounted for by the phonology of Parser 1 simply by removing the accent marking of the gold standard transcriptions. Form (44a) is an example of such a word. The phonology of Parser 1 does not generate *iihpómaawa*

²⁶¹On average across the five LM-dependent versions of Parser 1, the phonology can account for 67.4 of 325 words while the morphophonology can parse 63.4.

from /ohpommaa-wa/ because it contains no rules that insert non-lexical prominence marking. Instead, it generates the orthographic transcriptions in (44b,c). However, if accents were removed from orthographic transcriptions prior to parsing, then this phonology would be able to analyze this form and the parser would succeed at parsing it.

- (44) a. *iihpómmaawa*
ohpommaa-wa
buy-3SG
'she bought it' from (Frantz, 1991, p. 36)
- b. *iihpommaawa*
- c. *ohpommaawa*

However, simply removing prominence distinctions (i.e., accent diacritics) on transcriptions prior to parsing will not necessarily improve overall parser performance since doing so will cause the parser to fail on words where underlying lexical prominence surfaces in ways that *are* predicted by the phonology. Form (43) above is an example of a word that Parser 1 would fail to parse if the accent diacritics were removed.

The solution is to both remove accent diacritics from transcriptions prior to parsing *and* create a phonological rule that removes lexical accent diacritics. If modified in this way, Parser 1 will be able to parse both (43) and (44) above, as well as (45) below.

- (45) *áaksikahksstóókiyiwa*
áak-ikahk-sstooki-yii-wa
FUT-cut-ear-DIR-3SG
'he will cut its ear off' (Frantz and Russell, 1995, p. 34)

Now consider the gold standard well analyzed word in (46a). The phonology of Parser 1 fails to account for the phonemic-orthographic correspondence in this word. From the morphophonemic form /nit-iik-waakomimm-ok-yini/ it generates the four transcriptions in (46b-e). In this case, removing accent diacritics from the phonemic morpheme transcriptions and the orthographic word transcriptions will not help things since there are changes in vowel length that the phonology does not predict. However, if long vowels and consonants were reduced to their short counter parts, if accented vowels were reduced to their unaccented counterparts, and if both of these transformations were performed both on morpheme representations by the phonology and on the orthographic transcriptions prior to parsing, then the morphophonology of the parser would be able to generate correct candidates for an even larger portion of the data set.

- (46) a. *nitsikákomimmokini*
nit-iik-waakomimm-ok-yini
1-INT-love-INV-4SG
'she loves me' (Frantz, 1991, p. 56)
- b. nitsiikaakomimmokini
c. nitsiikaakomimmoksini
d. nitsiikaakomimmookini
e. nitsiikaakomimmooksini

The results of implementing the two strategies just discussed (i.e. removing prominence and length distinctions, both lexically and orthographically) are explored in the discussion of Parser 2 in § 5.3. One thing to consider is that such an approach causes the morphophonology to over-generate to an even greater degree than it does already. The morphophonology of Parser

1 already generates, on average, 7.8 orthographic transcriptions for each morphological analysis.²⁶² One consequence of this over-generation is that the burden of selecting the correct parse is shifted to the LM, such that improvements to the LM will be more significant to the overall success of the parser. Another consequence is that the parser becomes less useful as a generator; that is, if one wants to use a morphological parser to generate a list of orthographic/phonetic representations of grammatical words of Blackfoot, then this approach will generate many ungrammatical forms.²⁶³

While the approach just described does improve the overall performance of the parser (cf. § 5.3.4), it is, for the reasons just pointed out, problematic and, therefore, a better approach would involve improving the phonology so that it can accurately model the introduction of non-lexical pitch accent in Blackfoot words. At this point, I leave to future work the task of improving the phonology in this way by incorporating the generalizations and analyses of works such as Weber (2013), Weber and Allen (2012), and Kaneko (1999).²⁶⁴

The parser described here was created with the goal of faithfully mod-

²⁶²This figure is an average over the set of words for which one or more candidate analyses were generated. Only 44% of parse attempts resulted in the production of one or more parse candidates. The average number of candidates generated per parse attempt is 3.4.

²⁶³It might, however, be feasible to train LMs on character sequences so that the probabilities of orthographic and phonetic representations could be estimated and, consequently, so that a generator built on such a parser could output a single most probable surface representation for each possible sequence of morphemes. This is a possibility I leave for future work.

²⁶⁴Incorporating these pitch accent location generalizations will require creating phonologies that can syllabify Blackfoot words, a task that will be made easier by the work of Denzer-King (2009). In addition, the fact that the location of accent on Blackfoot words is sensitive to morphological context (cf. Weber, 2013) will furnish an opportunity to demonstrate the usefulness of a *rich upper* morphophonology FST (cf. § 3.4).

elling the lexicon of Frantz and Russell (1995) and the phonology and allomorphy of Frantz (1991). While certain shortcuts were taken (cf. § 5.2.1), the resulting morphophonology FST constitutes a reasonably faithful computational implementation of the analyses just mentioned and, as such, it may be used as a discovery tool for linguistic research. The previous discussion has illustrated this by using the performance of the parser to support what works such as Weber (2013) have already claimed, viz. that a major gap in the phonology of Frantz (1991) is the lack of an analysis of non-lexical pitch accent. Once the phonology is improved in this way, and once superficial inconsistencies in the data set (cf. § 4.3) are factored out, the resulting parser and morphophonology FST may help to reveal further gaps in the analysis.

5.3 Parser 2

Parser 2 is Parser 1 with a phonology that removes prominence and length distinctions in generation. The result of this is that many possible morpheme sequences will generate a given surface (orthographic) representation. This places a greater burden on the LM. Consequently, the parser is evaluated with the LM of Parser 1 (§ 5.2.3) as well as with a categorial one described in § 5.3.3 below.

One issue with Parser 2 is that it is slow. The reason for this is that its morphophonology (on account of its distinction-obliterating phonology) produces approximately 1,800 morphological analysis candidates for any given input. The LM must then compute the probability of each candidate and sort them accordingly. On my machine, Parser 2 requires an average of 0.22

seconds to parse any given word. Given that a primary use case of these parsers is to produce analyses during data entry into a fieldwork tool like the OLD, this suboptimal parse speed may be an issue. However, the fact that OLD parsers cache their parses reduces the seriousness of this issue. Once parsers are incorporated into the in-development GUI of the OLD fieldwork tool, the system could be configured to run a parser on previously unparsed words when users are not active, thus building the parse cache and facilitating greater responsiveness from the user's point of view.

5.3.1 Phonology

The phonology of Parser 2 is identical to that of Parser 1 except for the introduction of two simple rules that apply after all the others: *no accented vowels* and *shorten*.²⁶⁵ The first (Figure 5.47) converts accented vowels to their unaccented counterparts while the second (Figure 5.48) reduces long consonants and vowels to their short counterparts.

```
define noAccentedVowels á -> a , í -> i , ó -> o ;
```

Figure 5.47: No accented vowels.

²⁶⁵The complete foma phonology script for Parser 2 can be found on the GitHub repository of the `old-parser-research` tool, i.e., at <https://github.com/jrwdunham/old-parser-research>; see the file `blackfoot_phonology_frantz91_flattener.script` in the `resources/` directory.

```
define shorten [  
  p+ @-> p ,  
  t+ @-> t ,  
  k+ @-> k ,  
  m+ @-> m ,  
  n+ @-> n ,  
  s+ @-> s ,  
  a+ @-> a ,  
  i+ @-> i ,  
  o+ @-> o ] ;
```

Figure 5.48: Shorten.

Note the use of the left-to-right longest-match replacement operator @-> in Figure 5.48 (Hulden, 2012). This ensures that the longest contiguous sequence of a given character is replaced with one instance of that character.

5.3.2 Morphology

The morphology of Parser 2 is the same as that of Parser 1. See § 5.2.2 above.

5.3.3 Language Model

Parser 2 is evaluated with two LMs. The first is the LM of Parser 1. For a description of that LM, see § 5.2.3 above. The second is identical to that of Parser 1 but is categorial. That is, the trigrams that are encoded in this second LM are sequences of morpheme categories and not sequences of morphemes (i.e., (*morpheme break*, *morpheme gloss*, *syntactic category*) triples).

The second LM of Parser 2 is estimated from the set of 3,245 gold stan-

dard well analyzed word types, which is described in § 4.2.2 above. As with the non-categorial LM of Parser 1, five versions of this LM were created, each using a different random sample of 90% of the gold standard well analyzed words as training data. The test sets consisting of the remaining 10% were used to evaluate the parsers built atop the appropriately trained LMs.

The second LM of Parser 2 is *categorial*, has order=3, and uses the modified Kneser-Ney (Hsu and Glass, 2008; Chen and Goodman, 1999) smoothing algorithm. This LM has 31 unigram, 270 bigram, and 820 trigram types. Table 5.3 shows the top 20 most frequent trigrams in one of Parser 2's five LMs.

5.3. Parser 2

category trigram	count
thm-agrb#	227
vai-agrb#	210
#agra-adt	191
adt-vai#	177
#adt-adt	171
vrt-fin#	140
#adt-fin	129
#agra-vai	122
vta-thm-agrb	113
nan-num#	110
agrb-oth#	109
nin-num#	103
agra-vai#	100
adt-fin#	100
#adt-vai	99
adt-vrt-fin	87
#asp-vai	86
#agra-nar	84
#nan-num	83
asp-vai#	79

Table 5.3: 20 most frequent category trigrams in one of Parser 2’s training LMs.

As Table 5.3 indicates, the most frequent trigrams of the categorial LMs have more counts than their morpheme-based counterparts in Table 5.1. In general, a category-based N -gram LM has fewer possible trigrams and higher counts for each trigram. Any two analyses with the same category sequence will be assigned the same probability by a category-based LM and the system will arbitrarily choose one as the most probable. Since the morphophonology of Parser 2 generates approximately 1,800 analyses per

word transcription, there is a very real possibility that multiple candidate analyses will have the same category sequence. This may explain the fact that the use of a category-based LM causes Parser 2 to perform more poorly.

5.3.4 Results

Table 5.4 shows the performance measures of Parser 2 and repeats the results of Parser 1 from Table 5.2 for easy comparison. *Parser 2a* refers to the version with the morpheme LM described in § 5.2.3 while *Parser 2b* refers to the version with the categorial LM described in § 5.3.3.

parser	parse rate	F-score	precision	recall	phonology	morphon	LM
1	0.14	0.32	0.53	0.23	0.21	0.20	0.72
2a	0.17	0.40	0.40	0.39	0.60	0.60	0.28
2b	0.15	0.38	0.41	0.36	0.60	0.60	0.24

Table 5.4: Parser 2 results.

Both versions of Parser 2 perform better than Parser 1, both in terms of F-score and overall parse success rate. With a 60% success rate, the morphophonology of Parser 2 performs three times better than its counterpart in Parser 1. However, the LM success rate has decreased from 0.72 to 0.28 and 0.24 for Parsers 2a and 2b, respectively, thus explaining the modest (3% and 1%) increases in overall parse rate. Because the phonology obliterates prominence and length contrasts, now not only does the morphophonology FST recognize many more transcriptions, it also produces many more candidate parses per transcription. The approximate figure is 1,800 parse candidates for any given transcription. The LMs are, therefore, tasked with ranking a

significantly larger set of candidates and their inadequacy, which is a result of their relatively small training sets (cf. § 5.2.3 and § 5.3.3 above), is beginning to show itself. The categorial LM of Parser 2b actually performs less well than the morpheme LM of Parsers 1 and 2a. The likely reason for this is that, because of the high number of parse candidates returned by the morphophonology component, it is likely that multiple candidates will have the same category sequence and thus the category-based LM will not be able to select a most probable parse.

Parser 2a is the best parser created in the present study. It is interesting to note that even though it returns the correct parse only 17% of the time, *a* parse was returned 88% of the time. With a precision value of 40%, this means that for 88% of words entered by a user, Parser 2 will return a suggested analysis such that 40% of the morphemes it contains are, on average, correct. While not ideal, this would still be helpful to fieldworkers since editing a half-correct suggestion should be less work than specifying a correct one from scratch. Additionally, since the morphophonology contains the correct analysis 60% of the time, a GUI that mediates interaction with this parser could allow the user to choose from a list of analyses ordered by probability. Finally, as more gold standard well analyzed data are created (as expedited by the parser itself), the LM can be progressively improved, thus improving the performance of the parser as a whole.

5.3.5 Discussion

This section reviews a random sample of 20 gold standard words that Parser 2 is unable to account for. Inspection of these failures reveals possibilities

for improving the phonology, issues with the flattening approach taken, and problems with the data set. Note that this exploration of the data points that this parser and its underlying morphophonological models cannot account for is an act of linguistic research, one that is made possible by the computational implementation of these models and the automatic evaluation of these models against the data sets, as provisioned by the OLD. This discussion thus constitutes an illustration of one way in which the OLD can facilitate linguistic research.

Five of the 20 phonological failures were due to omissions of word-final devoiced vowels in orthographic transcriptions, as illustrated by (47a,b).²⁶⁶ Since this word-final vowel omission phenomenon caused the phonology to fail on 25% of the sample, adding an optional phonological rule to this effect would probably improve the overall performance of the parser.

- (47) a. *itáísaksíaw*
 it-á-saksi-yi-aawa
 LOC-IMPF-exit-3PL-PRO
 prev-asp-vai-agrb-oth
 ‘they went out’ (Uhlenbeck, 1911)
- b. *kitaaksaaksoy*
 kit-áak-sa’kssoyi
 2-FUT-burn.oneself
 agra-ten-vai
 ‘you will burn yourself’ (JD, 2005/10/11)

Note that (47b) illustrates another phenomenon—also seen in (48)—where glottal stops are omitted from transcriptions (or are not produced by

²⁶⁶The non-standard transcription—accents on the first but not the second syllable of the diphthong/coalesced vowel digraph <ai>—of (47a) is faithful to Uhlenbeck (1911). The morphological analysis, however, is my own.

speakers).

- (48) *akohkoottapootaki*
áak-ohkott-á'p-o't-aaki
FUT-able-about-grasp-PS.INTR
ten-mod-adt-vrt-fin
'he can work' (MW, 2005/06/28)

On the other hand, the predictable glottalization of sonorants before voiceless vowels is sometimes transcribed, as shown in (49).

- (49) *kian'ohk*
ki-annohk
and-now
und-und
'and now' (JD, 2005/10/11)

The sample set of 20 phonological failures reveals some lexical items that participate in, but were not accounted for in, the allomorphic alternations specified in § 5.2.1 above. Given data like (50), the become-/o/ alternation (cf. Figure 5.34) should perhaps be modified to include reference to the morpheme /apak/ 'wide'.

- (50) *ikoopaksikimii*
iik-apak-ikim-yi
INT-wide-liquid-be
adt-adt-med-fin
'it (river) is wide' (JD, 2008/05/22)

Similarly, the failure of the phonology on (51) indicates that the adjunct /mikksk/ 'brittle' is /y/-initial in verbal forms but /m/-initial in nominal

forms, a phenomenon that should probably be accounted for in the nasal-initial verbs rule of Figure 5.30 or some other lexical phonological rule.

- (51) *iiksskáí'yikkskiwa*
 iik-sska'-mikksk-yi-wa
 INT-greatly-brittle-be-3SG
 adt-adt-adt-fin-agrb
 'it is extremely brittle' (Frantz and Russell, 1995, p. 126)

Data like (52) indicate that the lexical phonological rule which deletes word-initial /i/ in imperatives (cf. Figure 5.31 above) should probably have its environment of application generalized to include nominalizations.²⁶⁷

- (52) *paapáówahsin*
 ipapa-aoówahsin
 in.dream-food
 adt-nin
 'popcorn' (Frantz and Russell, 1995, p. 69)

Finally, data like (53) indicate that perhaps nasals should be included in the set of segments that trigger /oh/-epenthesis (cf. Figure 5.33 above).

- (53) *noohko's*
 n-ko's
 1-dish
 agra-nan
 'my dish' (JD, 2008/07/08)

²⁶⁷It is unclear whether the word /aoówahsin/ 'food' is synchronically a nominalization (via /-hsin/ suffixation) of /ooyi/ 'eat', /oowat/ 'eat' (*vta*), or some putative /oo/ 'eat' (*vrt*). Since there are /i/-initial mono-morphemic noun stems that retain their left-edge /i/, the generalization may be that /i/-initial adjunct modifiers lose their initial vowel when prefixed to nouns but not when prefixed to verbs. Capturing this generalization, however, would probably require a *rich upper* morphology where the phonology could have access to categorial information.

On the other hand, the phonology should perhaps *not* be modified to account for the data points in (54), which are probably misanalyzed.

- (54) a. /ann-wa/ → <annay>
 b. /á-ikk-atw-imaa-yi-aawa/ → <aikatsimaya'wa>
 c. /it-oto-isam-sootaa/ → <ito'tsisamsota>
 d. /káák-oht-ooto-ssamm-yii-wa-áyi/ → <kakohtotaisamiwayi>
 e. /kit-áíssksinima'tstohki-ooawa-yi-aawa/ →
 <kitaíssksinima'tstohkoayawa>
 f. /kit-áák-á'p-o't-aaki-hsi/ → <kitaka'po'takhi>
 g. /it-yaamaahki-aaki/ → <itsamahkyiaki>

Turning to the failures that are due to Parser 2's LM, consider that for the orthographic word <á'paistotakiwa> 'she is making it', Parser 2 should have returned the analysis in (55). However, (56) is what Parser 2 actually estimated as most probable. In fact, Parser 2 ranked 15 analyses as more probable than the correct analysis. Parser 2's next four most probable candidate analyses are given in (57a–d). The important thing to note here is that the LM ranker would never be forced to rank analyses containing the morpheme /ottaki/ 'dip out liquid' if it were not for the fact that Parser 2's phonology obliterates segment length distinctions. This example highlights the need for a more accurate phonology, one which can predict non-lexical prominence location and *real* segment length alternations.

- (55) á'paistotakiwa
 á'p-á-istot-aaki-wa
 about-IMPF-make-PS.INTR-3SG
 adt-asp-fin-fin-agrb
 'she is making it' (Frantz and Russell, 1995, p. 18)

- (56) *á'p-waist-ottaki-wa*
about-near.speaker-dip.out.liquid-3SG
adt-adt-vai-agrb
- (57) a. *á'p-aist-ottaki-wa*
about-to.speaker-dip.out.liquid-3SG
adt-adt-vai-agrb
- b. *á'p-á-aist-ottaki-wa*
about-IMPF-to.speaker-dip.out.liquid-3SG
adt-asp-adt-vai-agrb
- c. *á'p-á-ist-ottaki-wa*
about-IMPF-two-dip.out.liquid-3SG
adt-asp-adt-vai-agrb
- d. *á'p-aist-ottaki-wa*
about-to.speaker-dip.out.liquid-PROX.SG
adt-adt-vai-num

The construction of Parser 2 has been revealing. With a phonology that removes prominence and length distinctions (§ 5.3.1), it is possible to build a morphophonology FST that can recognize and produce candidate parses for 60% of the gold standard well analyzed words of the Blackfoot data set. By further modifying the phonology to account for the cases where it was revealed to have failed (cf. (47-53)), it should be possible to further improve its success at mapping the correct analysis to the correct transcription. However, given that the data set is inconsistent in a number of ways (cf. § 4.3 and the examples in (54)), a morphophonology that is able, via further proliferation of candidate analyses, to analyze the bulk of this data set is probably not desirable. This is because such a morphophonology would, as (55-57) shows, impose an unmanageable burden on the small LMs tasked with ranking the vast candidate sets that it would produce.

Future work on creating better performing Blackfoot morphological parsers will follow a two-pronged approach. On the one hand, the existing parsers can be used to expedite the creation of larger corpora of morphologically well analyzed words which can, in turn, be used to estimate more accurate LMs.²⁶⁸ On the other hand, development of a more accurate phonology, informed by the data points discussed here and in § 5.2.5, will reduce the burden on the LM candidate ranker.

5.4 Discussion

This section has described and evaluated two morphological parsers for the Blackfoot language that were constructed using the OLD’s morphological parser creator application (chapter 3). Though evaluated against a highly inconsistent data set (§ 4.3), the best performing parser is able to correctly parse 17% of the test corpus and achieve an F-score of 0.4. Given that F-score and its ability to return *an* analysis for 88% of words, this parser can be used to suggest morphological analyses that linguists can edit, thus expediting the creation of more morphologically well analyzed data that can be used to further improve parse rates and which will allow for more sophisticated searches on the data set.

As shown above, inspection of the failures of parsers can help in uncov-

²⁶⁸The performance of the LM might also be improved simply by modifying the corpus from which it is estimated so that the forms instantiating the most common category sequences are replicated a number of times. A more sophisticated method of improving the LM—though one not yet made possible by the OLD—would be to use the relevant category-based *N*-gram counts to inform the estimation of morpheme-based trigram probabilities where morpheme *N*-gram counts are not available (i.e., category-informed back-off).

ering data points that are not predicted by their underlying models. These counterexamples range from the rather superficial—e.g., allomorphs of morphemes that were simply overlooked in lexical phonological rules—to the more interesting—e.g., non-lexical prominence patterns and possible glottal stop elision. Though not explored here, parsers could also be used as an analytical discovery tool in a more direct way, i.e., by building a phonology that implements a particular analysis of a phenomenon and then searching through the data points that are not captured by the phonology. For example, by writing a phonology encoding a particular analysis of vowel hiatus resolution in Blackfoot, and by then searching through the unaccounted for data points for vowel-hyphen-vowel sequences in the morpheme break value, one could quickly find counterexamples to the analysis in question.²⁶⁹

This experiment in creating morphological parsers for Blackfoot has made it clear that there are pressing issues related to the orthography and phonology of the language and, further, that parsers and their components may have an effect on how these issues are addressed. In order for an orthographic parser to be practical, it is necessary to decide whether orthographic transcriptions should reflect the phonetics of individual pronunciations, an idealized pronunciation (perhaps that of a specific individual or dialect), or a phonemic abstraction. The standard orthography of the language (Frantz, 1991, 1978) lies, in fact, somewhere between the first two options: it is phonetic to a certain degree and also reflective of speaker-specific variability. Yet it is unclear whether phenomena such as word-final devoicing and nasal

²⁶⁹Thanks to Naoki Peter for discussing with me the use of OLD morphological parsers for this purpose.

glottalization should be indicated orthographically (cf. § 4.3.3). The phonology (i.e., spelling rules) of an orthographic Blackfoot parser must address these issues and decisions made may, in turn, (if these parsers and their components see practical use) affect the wider orthographic conventions.

Relatedly, since some linguists deal in phonetic IPA transcriptions of various degrees of granularity, it may be useful to create *phonetic* parsers, potentially phonologically permissive ones as well as speaker-specific ones. These could be used to parse as well as to *generate*, i.e., from morpheme break values. In this way, one could enter Blackfoot data orthographically, phonetically, or morphophonemically and have the other representations auto-generated based on specified parsers and their morphophonology FSTs. Such true phonetic parsers could also facilitate more rigorous testing (as compared to the orthographic parsers described here) of generalizations and analyses.

Future work on Blackfoot morphological parsers based on the models defined here may also include the development of spell-checkers and the generation of pronunciation dictionaries for use in software that automates the alignment of transcriptions and audio recordings, e.g., Prosodylab-Aligner (Gorman et al., 2011).

Chapter 6

Conclusion

This work has described the Online Linguistic Database (OLD) and has argued that it is software that effectively facilitates linguistic fieldwork. The fieldwork chapter (2) described and argued for the primary features of the system and compared it to similar tools. The OLD is an open-source, platform-agnostic, web-based application that helps fieldworkers to share data and work collaboratively. Its web service-based architecture facilitates the reuse of its data stores and of its functionality. Its data structure arose from the author's own fieldwork experience and has evolved in response to feedback from numerous other fieldworkers and multiple field methods classes. It allows for sophisticated searches (including regular expression and structural searches) across multiple fields, including system-generated serializations of morphological analyses. Its permissions system allows administrators to control who has what level of access to which data. It promotes consistency with features like orthography conversion, inventory-based input validation, and visual feedback on lexical consistency of morphological analyses. Within the system, one can create and export formatted documents that contain data points from within the database.

The morphological parser creator is an application added to the core of

the OLD. Chapter 3 detailed how to use the parser creator to build morphological parsers and how to implement morphological and phonological models as FSTs. The parsers and their components can be used to automate the creation of morphological (IGT) representations and to evaluate the observational adequacy of analytic models against specified data sets. Chapter 5 illustrated the use of the parser creator in the building of parsers for Blackfoot, using an OLD data set described in chapter 4. This experiment showed how parsers can be created which are practical tools for expediting data creation and for automating analysis evaluation.

The OLD is an advantageous tool for linguistic fieldwork. This claim is supported by the simple fact that OLD applications constitute coherent multi-contributor linguistic artifacts that are potentially valuable to documentation, research, and revitalization efforts, perhaps even in eventualities where empirical work is no longer possible. The many features of the software, as described and evaluated above, further support the value of the OLD as a fieldwork tool. The usefulness of the OLD is itself support for the general methodological claim that it is sound strategy to increasingly turn linguistic resources toward engineering effective tools that promote speedier creation and increased and improved reuse of waning endangered language data.

The OLD aims to facilitate better linguistic research by doing more than simply enabling the sharing of data and providing efficiency-promoting conveniences. By implementing an interface for performing advanced searches on consistent, well-structured, and analyzed data, the OLD promises to allow researchers to discover patterns that might otherwise have remained

hidden. Also, by allowing users to model components of the grammar and to embed these components within a dynamic multi-contributor data set, the resulting models can be efficiently tested against large data sets and consequently rendered more explicit and observationally adequate than would otherwise be the case. As shown in § 5.3.5 above, by returning data points that a given model cannot account for, the OLD can assist in the discovery of phenomena that are relevant to linguistic research.

Since the OLD can benefit linguistic research, the *development* of the OLD (and similar tools) is itself a valuable contribution to linguistic research. This does not entail that *all* field linguists must develop advanced computational linguistic and/or programming skills. It simply suggests that linguists might want to consider broadening their view of the skill set that is valuable to the field and, consequently, diversifying recruitment and increasing the division of labour so that theoreticians, fieldworkers, programmers, applied linguists, and computational linguists can coordinate efforts towards the achievement of common goals.

The remainder of this conclusion discusses some exciting potential improvements to the OLD that will further facilitate the accomplishment of fieldwork-related goals. In particular, it discusses improvements to the user interface, more analysis of the metalanguage, the automatic alignment of audio with transcriptions, the creation of functionality for implementing syntactic models, the use of OLD applications and data in language-learning software, the publicization of OLD data, and the use of OLD applications for cross-linguistic and historical linguistic research.

As already discussed, there is still much work to do on the OLD user in-

terface. The present text has focused primarily on describing and evaluating the OLD web service, However, there is at present no graphical user interface for this repurposable tool (cf. Table 2.3). Some salient challenges for such an interface will include a search form that can allow users to construct complex hierarchical searches using regular expressions and structural (TGrep2) regular expression patterns (cf. § 2.3.6). An effective approach to designing an interface for building a tree structure of coordinated and negated filter expressions might be to have the system allow the user to assemble a set of filter expressions in such a way that they actually do form a visual representation of a tree structure. In the case of building complex regular expressions, it would be useful to allow users to define macros, i.e., variables that reference regular expression patterns, and then to use these in order to more transparently build more complex expressions. For example, by allowing users to define reusable macros for regular expressions that match consonants, vowels, and syllables, we could facilitate the creation of search queries that target specific patterns in particular (locations of particular) syllables. Other useful GUI-related innovations would include real-time rendering of formatted OLD texts (i.e., collections) *during* their composition via a lightweight markup language (i.e., reStructuredText or Markdown), real-time construction of graphical representations of syntax trees *while* they are being specified via bracket notation, alternative views (e.g., tabular) of form data, and a generally improved user experience, e.g., keyboard shortcuts, user-customization, removal of page refreshes, etc.

The OLD can, in its current state, be useful to fieldworkers who are primarily concerned with the learning, teaching, and revitalization of endan-

gered languages. However, as is indicated by the focus on creating morphological parsers and performing other research-related tasks such as searches over analytical representations, the development of the OLD has thus far been focused on research-relevant goals to over revitalization-relevant ones. At present, revitalizer-fieldworkers can construct (rough drafts of) lessons or other learning materials using the OLD's functionality for creating texts with embedded representations of form data. Learners can also use the dictionary and search interfaces to look up words, morphemes, and translations of English forms. As OLD data sets grow in size and become increasingly consistent, accessible, and rich (i.e., with multiple levels of analysis and annotation and with aligned audio/video data), their potential value to language learning and related endeavours increases. I envision using OLD data sets to construct dynamic corpora that can feed structured data to talking (i.e., audio/textual) dictionaries and language-learning software. There is, in my estimation, a lot of potential to produce high quality language-learning web applications that make use of OLD data and APIs.

Another interesting potential improvement to the OLD would involve allowing contributors to analyze the metalanguage data (i.e., the translations) and to build tools to automate such analysis. Since the metalanguage is often English, Spanish, or some other majority language which has already been the object of significant computational linguistic engineering efforts, there are many existing tools (e.g., syntactic and morphological parsers) for building such analyses. By incorporating analyses of the metalanguage into an OLD data set, search functionality could be improved. This is especially true in syntactic and semantic research where it is often useful to be able

to search through language data based on structural or semantic cues in its possible translations.

Audio (and video) recordings of endangered language utterances that are time-aligned with appropriate transcriptions and analyses are highly valuable. For researchers, it is very helpful to be able to listen to how a form was produced (by a speaker, but also by an elicitor) in order to verify and assess the corresponding transcriptions and analyses. For revitalizers and software developers building language-learning software, such aligned data are also highly valuable when it comes to creating multi-media artifacts. However, creating such data is a highly time-consuming process. Therefore, a much-anticipated future development will seek to make use of an open-source tool called the Prosodylab-Aligner which can time-align transcriptions of speech with audio recordings thereof (Gorman et al., 2011). In order to function, this tool requires a pronunciation dictionary listing all of the words (with phonetic representations) that are used in the transcriptions. Conveniently, since it can be used to computationally implement models of the morphology and phonology of a language, the OLD's morphological parser creator promises to be highly useful in the creation of such pronunciation dictionaries.

Another exciting addition to the OLD would be the creation of functionality to allow users to implement analyses and models of the syntax. Similar to the morphophonological modelling described and evaluated above, this would facilitate baseline testing of syntactic analyses against large and select data sets as well as the creation of functional syntactic parsers, which would in turn assist in the more rapid generation of phrase structure (or de-

pendency) representations which could in turn be structurally searched via the OLD's existing tree search interface (i.e., TGrep2), thus increasing the value of the OLD with respect to linguistic research. The implementation of syntactic models and the creation of syntactic parsers could, if needed, be furthered by building upon the morphological representations generated via the the morphological parser functionality. In addition, syntactic representations could form the basis for implementing formal semantic analyses and automatically generating semantic representations. It would be useful to be able to search for patterns in the semantic representations of object language data and to further filter these queries according to restrictions on the semantics of the corresponding metalanguage translations.

While the OLD's RESTful/JSON architecture facilitates data reuse, there are still a number of obstacles. In particular, the access restrictions of the system currently do not allow for (portions of) OLD data to be made unrestrictedly public, i.e., available without password-based authentication. Relatedly, work also needs to be done on making it easier for OLD users to easily transfer their data sets to language archives. Another related innovation would involve making OLD data even more easily reusable by exposing it in adherence to Linked Open Data standards (Berners-Lee et al., 2009; Heath and Bizer, 2011).

A final exciting set of improvements to the OLD would involve making the system more usable for cross-linguistic and historical linguistic research. To the first point, it would not be difficult to modify the existing code base to produce a system that implements a single interface to multiple language-specific OLD instances. Such an interface could allow for sophisti-

cated cross-linguistic searches and facilitate the creation of research papers that reference data from multiple languages. To the second point, the data structure of the system could be extended to provide support for reconstructed forms, cognacy judgments, etc.

There are clearly many exciting and useful improvements that could be made to the OLD. I hope the system in its current form will be useful to linguists, fieldworkers, and community members and I hope that this work will result in improvements to the system and, more importantly, that it may inspire others to pursue and develop innovations in the domain of computationally-assisted collaborative linguistic fieldwork.

Bibliography

- Afitska, O. (2009). Review of Transana 2.30. *Language Documentation & Conservation*, 3(2):226–235.
- Amery, R. (2009). Phoenix or relic? Documentation of languages with revitalization in mind. *Language Documentation & Conservation*, 3(2):138–148.
- AnderBois, S. and Henderson, R. (to appear). Linguistically establishing discourse context: two case studies from Mayan languages. In Bochnak, R. and Matthewson, L., editors, *Methodologies in Semantic Fieldwork*, pages 207–233. Oxford University Press.
- Armoskaite, S. (2006). Heteromorphemic assibilation of *k* in Blackfoot. UBC, qualifying paper.
- Armoskaite, S. (2011). *The destiny of roots in Blackfoot and Lithuanian*. PhD thesis, University of British Columbia.
- Barwick, L. (2009). Review of Transcribe! *Language Documentation & Conservation*, 3(2):236–240.
- Beale, S. (2012). Documenting endangered languages with Linguist’s Assistant. *Language Documentation & Conservation*, 6:104–134.

- Beermann, D. and Mihaylov, P. (2010). Cloud computing for linguists. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 49–52. Association for Computational Linguistics.
- Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology*. Palo Alto, CA: CSLI Publications.
- Berez, A. L. (2007). Review of EUDICO linguistic annotator (ELAN). *Language Documentation & Conservation*, 1(2):283–289.
- Berners-Lee, T. et al. (2009). Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22.
- Bickel, B., Comrie, B., and Haspelmath, M. (2008). The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses. <http://www.eva.mpg.de/lingua/resources/glossing-rules.php>. Accessed: 2014-07-28.
- Bird, S., Chiang, D., Frowein, F., Berez, A. L., Eby, M., Hanke, F., Shelby, R., Vaswani, A., and Wan, A. (2013). The International Workshop on Language Preservation: An experiment in text collection and language technology. *Language Documentation & Conservation*, 7:155–167.
- Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O’Reilly Media.
- Black, H. A. and Simons, G. F. (2008). The SIL FieldWorks Language

Bibliography

- Explorer approach to morphological parsing. *Computational Linguistics for Less-studied Languages: Proceedings of Texas Linguistics Society*, 10.
- Bliss, H. A. (2013). *The Blackfoot configurationality conspiracy: Parallels and differences in clausal and nominal structures*. PhD thesis, University of British Columbia.
- Boas, F. (1917). Introductory. *IJAL*, 1:1–8.
- Bochnak, R. and Matthewson, L., editors (to appear). *Methodologies in Semantic Fieldwork*. Oxford University Press.
- Bowern, C. (2007). Review of TshwaneLex dictionary compilation software. *Language Documentation & Conservation*, 1(1):94–99.
- Bowern, C. (2008). *Linguistic Fieldwork: a practical guide*. Palgrave Macmillan.
- Brotchie, A. (2007). Review of Audiamus 2.3. *Language Documentation & Conservation*, 1(2):290–292.
- Buchan, H. (2011). Review of Phon: Free software for phonological transcription and analysis. *Language Documentation & Conservation*, 5:81–87.
- Butler, L. and van Volkinburg, H. (2007). Review of Fieldworks Language Explorer (FLEx). *Language Documentation & Conservation*, 1(1):100–106.
- Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Bibliography

- Chomsky, N. (1965). *Aspects of the Theory of Syntax*, volume 11. The MIT press.
- Chomsky, N. (1980). Rules and representations. *Behavioral and Brain Sciences*, 3(01):1–15.
- Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper & Row, New York.
- Churchill, W. (2004). *Kill the Indian, save the man: The genocidal impact of American Indian residential schools*. City Lights Books, San Francisco.
- Crippen, J. (2010). Bits, bytes, and unicode: An introduction to digital text for linguists. <http://www.drangle.com/~james/papers/bits-bytes-unicode.pdf>. Accessed: 2014-01-03.
- Crockford, D. (2008). *JavaScript: The good parts*. O'Reilly Media, Inc.
- Davis, H. (2012). A Teacher's Grammar of Upper St'át'imcets. manuscript, University of British Columbia, Vancouver, B.C.
- Davis, H., Gillon, C., and Matthewson, L. (2014). How to investigate linguistic diversity: Lessons from the Pacific Northwest. *Language*, to appear.
- Déchaine, R.-M. and Wiltschko, M. (2010). Micro-variation in agreement, clause-typing and finiteness: Comparative evidence from Blackfoot and Plains Cree. In *The Proceedings of the 42nd Algonquian Conference*.
- Denzer-King, R. E. (2009). The distribution of /s/ in Blackfoot: An optimality theory account. Master's thesis, University of Lethbridge.

Bibliography

- Derrick, D. (2006). Syllabification and Blackfoot ‘s’. Generals Paper, UBC.
- Dimmendaal, G. J. (2001). Places and people: field sites and informants. In Newman, P. and Ratliff, M., editors, *Linguistic Fieldwork*, chapter 3, pages 55–75. Cambridge University Press, Cambridge.
- Dobrin, L. M. (2009). SIL International and the disciplinary culture of linguistics: Introduction. *Language*, 85(3):618–619.
- Dobrin, L. M. and Good, J. (2009). Practical language development: Whose mission? *Language*, 85(3):619–629.
- Dorian, N. C. (1993). A response to Ladefoged’s other view of endangered languages. *Language*, pages 575–579.
- Dunham, J. (2013a). Old documentation (release 1.0a1). <https://online-linguistic-database.readthedocs.org>. Accessed: 2014-07-28.
- Dunham, J. (2013b). The Blackfoot Language Database. In Hele, K. S. and Valentine, J. R., editors, *Papers of the Forty-First Algonquian Conference*, pages 75–80, Concordia, Montreal.
- Dunham, J., Cook, G., and Horner, J. (2014). LingSync & the Online Linguistic Database: New models for the collection and management of data for language communities, linguists and language learners. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 24–33, Baltimore, Maryland, USA. Association for Computational Linguistics.

Bibliography

- Elfner, E. J. (2006). The mora in Blackfoot. Master's thesis, University of Calgary.
- Epps, P. and Ladley, H. (2009). Syntax, souls, or speakers? on SIL and community language development. *Language*, 85(3):640–646.
- Farrar, S. (2010). Review of TypeCraft. *Language Documentation & Conservation*, 4:60–65.
- Fielding, R. (2000). *Representational state transfer: An architectural style for distributed hypermedia interaction*. PhD thesis, University of California, Irvine.
- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Rfc 2616. *Hypertext Transfer Protocol–HTTP/1.1*, 2(1):2–2.
- First Peoples' Heritage, Language and Culture Council (2010). Report on the Status of B.C. First Nations Languages. <http://www.fpcc.ca/files/PDF/2010-report-on-the-status-of-bc-first-nations-languages.pdf>. Accessed: 2014-07-28.
- Frantz, D. G. (1971). *Toward a Generative Grammar of Blackfoot*. Summer Institute of Linguistics Publications in Linguistics and Related Fields.
- Frantz, D. G. (1978). Copying from complements in Blackfoot. *Linguistic studies of native Canada*, pages 89–109.

Bibliography

- Frantz, D. G. (1991). *Blackfoot Grammar*. Toronto: University of Toronto Press.
- Frantz, D. G. and Russell, N. J. (1995). *Blackfoot Dictionary of Stems, Roots, and Affixes*. Toronto: University of Toronto Press.
- Galli, M., Soares, R., and Oeschger, I. (2003). Inner-browsing extending the browser navigation paradigm. https://developer.mozilla.org/en-US/docs/Inner-browsing_extending_the_browser_navigation_paradigm. Accessed: 2014-07-28.
- Garde, M. (2012). Review of InqScribe. *Language Documentation & Conservation*, 6:175–180.
- Gardner, J. (2008). *The Definitive Guide to Pylons*. Apress.
- Gick, B., Bliss, H., Karin, M., and Radanov, B. (2012). Articulation without acoustics: “soundless” vowels in Oneida and Blackfoot. *Journal of Phonetics*, 40:46–53.
- Gildea, D. and Jurafsky, D. (1996). Learning bias and phonological rule induction. *Computational Linguistics*, 22(4):497–530.
- GitHub (2014). Github — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/wiki/GitHub>. Accessed: 2014-07-28.
- Goddard, I. (1996). Introduction. In Sturtevant, W. C., editor, *Handbook of North American Indians: Languages*, volume 17, pages 1–16. Smithsonian Institution, Washington, D.C.

Bibliography

- Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.
- Good, J. and Cysouw, M. (2013). Languoid, doculect, and glossonym: Formalizing the notion ‘language’. *Language Documentation & Conservation*, 7.
- Gorman, K., Howell, J., and Wagner, M. (2011). Prosodylab-aligner: A tool for forced alignment of laboratory speech. *Canadian Acoustics*, 39(3):192–193.
- Hale, K., Krauss, M., Watahomigie, L. J., Yamamoto, A. Y., Craig, C., Jeanne, L. M., and England, N. C. (1992). Endangered languages. *Language*, pages 1–42.
- Hallett, D., Chandler, M. J., and Lalonde, C. E. (2007). Aboriginal language knowledge and youth suicide. *Cognitive Development*, 22(3):392–399.
- Handman, C. (2009). Language ideologies, endangered-language linguistics, and Christianization. *Language*, 85(3):635–639.
- Harrison, K. D. (2007). *When languages die: The extinction of the world’s languages and the erosion of human knowledge*. Oxford University Press.
- Heath, T. and Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: Theory and technology*, 1(1):1–136.
- Hellwig, B., Van Uytvanck, D., Hulsbosch, M., Somasundaram, A., Tacchetti, M., and Geerts, J. (2013). ELAN - Linguistic Annotator, ver-

sion 4.6.2. <http://www.mpi.nl/corpus/manuals/manual-elan.pdf>. Accessed: 2014-07-28.

Hsu, B.-J. and Glass, J. (2008). Iterative language model estimation: Efficient data structure & algorithms. In *Proceedings of Interspeech*, volume 8, pages 1–4.

Hulden, M. (2009). *Finite-state machine construction methods and algorithms for phonology and morphology*. PhD thesis, University of Arizona.

Hulden, M. (2012). foma: finite state compiler and C library (documentation). <https://code.google.com/p/foma>. Accessed: 2014-07-28.

International Phonetic Association (1999). *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press.

Johnson, C. D. (1972). *Formal aspects of phonological description*. Mouton, The Hague.

Jurafsky, D. and Martin, J. H. (2008). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, NJ, 2 edition.

Kaneko, I. (1999). A metrical analysis of blackfoot nominal accent in optimality theory. Master's thesis, University of British Columbia.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

- Karttunen, L., Kaplan, R. M., and Zaenen, A. (1992). Two-level morphology with composition. In *Proceedings of the 14th Conference on Computational Linguistics*, volume 1, pages 141–148. Association for Computational Linguistics.
- Kopka, H. and Daly, P. W. (2004). *A guide to LaTeX*. Addison-Wesley, 4 edition.
- Kotcheva, K. (2009). Review of LEXUS. *Language Documentation & Conservation*, 3(2):241–246.
- Krasner, G. E., Pope, S. T., et al. (1988). A description of the model-view-controller user interface paradigm in the Smalltalk-80 system. *Journal of Object-Oriented Programming*, 1(3):26–49.
- Krifka, M. (2011). Varieties of semantic evidence. In Maienborn, C., Portner, P., and von Stechow, K., editors, *Handbook of Semantics*, pages 242–267. de Gruyter, Berlin.
- Ladefoged, P. (1992). Another view of endangered languages. *Language*, pages 809–811.
- Lawson, B. and Sharp, R. (2011). *Introducing HTML5*. New Riders Publications.
- Leonard, W. Y. (2008). When is an “extinct language” not extinct. *Sustaining linguistic diversity: Endangered and minority languages and language varieties*, pages 23–33.

Bibliography

- Levinson, S. C. and Evans, N. (2010). Time for a sea-change in linguistics: Response to comments on ‘The myth of language universals’. *Lingua*, 120(12):2733–2758.
- Lewis, M. P., Simons, G. F., and Fennig, C. D., editors (2013). *Ethnologue: Languages of the World*. SIL International, Dallas, Texas, 17th edition.
- Lindley, L. and Lyon, J. (2012). 12 Upper Nicola Okanagan texts. *UBCWPL: Papers for ICSNL XLVII*, 32:173–246.
- LingSync (2013). LingSync: A fieldlinguistics database which adapts to its user’s I-Language (white paper). <https://github.com/OpenSourceFieldlinguistics/FieldDB/blob/master/docs/WhitePaper.pdf>. Accessed: 2014-07-28.
- LingSync (2014a). LingSync. <https://www.lingsync.org>. Accessed: 2014-07-28.
- LingSync (2014b). LingSync Dev Site. <https://www.lingsync.org/dev.html>. Accessed: 2014-07-28.
- Lowery, K. E. (2008). Review of Emdros: The database engine for analyzed or annotated text. *Language Documentation & Conservation*, 2(2):332–339.
- Lutz, M. (2013). *Learning Python*. O’Reilly Media.
- Lyon, J. and Greene-Wood, R. (2007). Lawrence Nicodemus’s Coeur d’Alene dictionary in root format. *University of Montana Occasional Papers in Linguistics*, 20.

Bibliography

- Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.
- Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.
- Martin, J. (1983). *Managing the data-base environment*. Prentice-Hall, Englewood Cliffs (NJ).
- Matthewson, L. (2004). On the methodology of semantic fieldwork. *International Journal of American Linguistics*, 70(4):369–415.
- Matthewson, L. (2005). *When I was Small: I Wan Kwikws: a Grammatical Analysis of St'át'imc Oral Narratives*. University of British Columbia Press.
- McConnell, S. (2004). *Code Complete*. O'Reilly Media, Inc.
- McElvenny, J. (2008). Review of Kirrkir. *Language Documentation & Conservation*, 2(1):160–165.
- Meakins, F. (2007). Computerized language analysis (CLAN). *Language Documentation & Conservation*, 1(1):107–112.
- Meißner, C. and Slavcheva, A. (2013). Review of EXMARaLDA. *Language Documentation & Conservation*, 7:31–40.
- Mithun, M. (2001). Who shapes the record: the speaker and the linguist. In Newman, P. and Ratliff, M., editors, *Linguistic Fieldwork*, chapter 2, pages 34–54. Cambridge University Press, Cambridge.

Bibliography

- MITLM (2013). MIT Language Modeling Toolkit Tutorial. <https://code.google.com/p/mitlm/wiki/Tutorial>. Accessed: 2014-07-28.
- Morey, S., Post, M. W., and Friedman, V. A. (2013). The language codes of ISO 639: A premature, ultimately unobtainable, and possibly damaging standardization. Handout of a talk given at the PARADISEC RRR Conference.
- Muehlbauer, J. (2005). Blackfoot morphology key. Unpublished.
- Murray, S. E. (2014). Reciprocity in fieldwork and theory. Currently under review for publication.
- Newman, P. and Ratliff, M. (2001). *Linguistic Fieldwork*. Cambridge University Press.
- Olson, K. S. (2009). SIL International: An emic view. *Language*, 85(3):646–658.
- Perlin, R. (2012). WeSay, a tool for collaborating on dictionaries with non-linguists. *Language Documentation & Conservation*, 6:181–186.
- Ringersma, J. and Kemps-Snijders, M. (2010). Reaction to the LEXUS review in the LD&C vol. 3, no. 2. *Language Documentation & Conservation*, 4:75–77.
- Robinson, S., Aumann, G., and Bird, S. (2007). Managing fieldwork data with Toolbox and the Natural Language Toolkit. *Language Documentation & Conservation*, 1(1):44–57.

Bibliography

- Roche, E. and Schabes, Y. (1997). *Finite-state language processing*. MIT press.
- Rogers, C. (2010). Review of FieldWorks Language Explorer (FLEX) 3.0. *Language Documentation & Conservation*, 4:78–84.
- Rohde, D. L. (2005). *TGrep2 User Manual version 1.15*. Massachusetts Institute of Technology.
- Schupbach, S. S. (2013). Situational demonstratives in Blackfoot. <https://arizona.openrepository.com/arizona/bitstream/10150/270993/1/schupbach.pdf>.
- Schützenberger, M. P. (1961). A remark on finite transducers. *Information and Control*, 4(2):185–196.
- SIL International (2013a). SIL fonts for downloading. http://scripts.sil.org/cms/scripts/page.php?cat_id=FontDownloads. Accessed: 2014-07-28.
- SIL International (2013b). Technical notes on FieldWorks send/receive. <http://fieldworks.sil.org/wp-content/TechnicalDocs/Technical%20Notes%20on%20FieldWorks%20Send-Receive.pdf>. Accessed: 2014-07-28.
- SIL International (2013c). The Linguist’s Shoebox: Understanding the Shoebox transition. <http://www.sil.org/computing/shoebox/TransQA.html>. Accessed: 2014-07-28.

- SIL International (2013d). Toolbox information. <http://www-01.sil.org/computing/toolbox/information.htm>. Accessed: 2014-07-28.
- SIL International (2014a). FieldWorks movies. <http://downloads.sil.org/FieldWorks/Movies/Demo%20Movies.html>. Accessed: 2014-07-28.
- SIL International (2014b). SIL FieldWorks Language Explorer (FLEx). <http://fieldworks.sil.org/flex/>. Accessed: 2014-07-28.
- Silva, M. A. R. (2008). Laryngeal specification in Blackfoot obstruents. UBC Linguistics PhD Qualifying Paper.
- Snoek, C., Thunder, D., Lõo, K., Arppe, A., Lachler, J., Moshagen, S., and Trosterud, T. (2014). Modeling the noun morphology of Plains Cree. *ACL 2014*.
- Stacy, E. (2005). Phonological aspects of Blackfoot prominence. Master's thesis, University of Calgary.
- Svelmoe, W. L. (2009). 'We do not want to masquerade as linguists': A short history of SIL and the academy. *Language*, 85(3):629–635.
- Tan, N. and Martin, J.-C. (2011). Review of ANVIL: Annotation of video and language data 5.0. *Language Documentation & Conservation*, 5:88–94.
- Taylor, A., Marcus, M., and Santorini, B. (2003). The Penn treebank: an overview. In *Treebanks*, pages 5–22. Springer.
- Taylor, A. R. (1969). *A Grammar of Blackfoot*. PhD thesis, University of California, Berkeley.

Bibliography

- Uhlenbeck, C. (1911). Original Blackfoot tales. *Verhandelingen der Koninklijke Academic van Wetenschappen te Amsterdam*.
- Unicode Consortium (2009). *The Unicode Standard, Version 5.2.0*. Unicode Consortium, Mountain View, CA.
- Van Der Mark, S. C. (2003). The phonetics of Blackfoot pitch accent. Master's thesis, University of Calgary.
- van Rijsbergen, C. J. (1975). *Information Retrieval*. Butterworths, London.
- Weber, N. (2013). Accent and prosody in Blackfoot verbs. http://www.academia.edu/4250143/Accent_and_prosody_in_Blackfoot_verbs. Accessed: 2014-07-28.
- Weber, N. and Allen, B. H. (2012). Blackfoot prominence: Insights from morpho-phonology. http://elk.library.ubc.ca/bitstream/handle/2429/43085/Weber_N_et_al_Blackfoot_pitch_accent_paper.pdf?sequence=1. Accessed: 2014-07-28.
- Wikipedia (2013). International Organization for Standardization — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/International_Organization_for_Standardization. Accessed: 2014-07-28.
- Wood, F., Archambeau, C., Gasthaus, J., James, L., and Teh, Y. W. (2009). A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1129–1136. ACM.

Bibliography

Woodbury, T. (2003). Defining documentary linguistics. *Language documentation and description*, 1(1):35–51.

Wycliffe (2013). Our beliefs. <http://www.wycliffe.org/About/WhatWeBelieve.aspx>. Accessed: 2014-01-08.

Yegerlehner, J. (1955). A note on eliciting techniques. *IJAL*, 21:286–288.

Appendix A

OLD 1.0 Data Structure

The data structure of the OLD is the way that the system organizes the data it stores. In database terminology this would be called the *schema*. This section describes the OLD data structure in detail.

The data in an OLD application are stored in a relational database. However, the data structure is here described using the language of objects, named attributes of those objects, and possible values for those attributes. The exposition begins with a précis on relational databases and how objects are stored inside of them. It then catalogues the objects and their attributes alphabetically. Included are specifications of what constitutes a licit value for each attribute as well as the methods of construction of system-generated values.

Note that the data structure described here is that of the current development version of the OLD 1.0. It differs from the data structure of the current production version 0.2.

Finally, note that identifier and date-time modified attributes are common to all models and are therefore described here in order to avoid repetition. The former is the integer value created by the system each time a new object is created. Each unique object of a given type has its own

unique integer identifier. The larger the identifier value the more recent the creation of the object. The date-time modified attribute holds a UTC timestamp generated by the application logic whenever an object is created or updated.

A.1 Relational databases

A relational database is a set of tables. Each OLD object has a corresponding table, e.g., the form table holds the data for all of the form objects. The columns of these tables are labelled and the labels correspond to the attributes of the objects, e.g., the form table has an orthographic transcription column and this corresponds to the orthographic transcription attribute of form objects. Each row in the table contains the data, i.e., the attribute values, for a unique object.

The word *relational* in relational database refers to the fact that the objects implicit in the database may have attributes whose values are determined by data in other tables. Consider the following simplistic example. Each form object has an elicitor and that elicitor has (at least) a first name and a last name. The non-relational approach to encoding this in a database table would be to create *elicitor first name* and *elicitor last name* columns in the form table. One problem with this approach is that it results in a lot of repetition: every form elicited by John Doe would require the strings *John* and *Doe* in the appropriate columns. Not only is this inelegant, it opens up the possibility of user error and inconsistency. Very soon there will be forms elicited by *Jonathan Doe*, *john doe*, *John M. Doe*, etc. Also consider how

very many edits might be required to do something that should be simple like changing the elicitor's first name from *John* to *Jonathan*. This data structure is doing a poor job of reflecting the real world fact that all of these values in all of these rows are really supposed to be referring to the same individual.

The relational approach is to create a person table, give each person row a unique identifier, and have the elicitor column of the form table contain references to the appropriate people using the identifiers. This way, the elicitor information is stored once and changes to elicitor data can be made easily and in one place. The type of relation between form and person tables just described is called a *many-to-one* relation since each form can have exactly one elicitor and multiple forms can have the same elicitor.

The relational approach also makes it much simpler to represent objects with attributes whose values are collections of other objects. Consider that a form should be able to have any number of tags and that multiple distinct forms should be able to have the same tag. We could simulate this data structure non-relationally via a table with a large number of tag columns. However, under this strategy the number of tags for each form would be arbitrarily bounded and the same criticisms levelled above would still apply. The relational approach is to create two new tables, a tag table to hold the tags and a form-tag table to encode the relations. Each row in the form-tag table contains one unique form identifier and one unique tag identifier. This type of relation is called *many-to-many* since it allows for a single form to have multiple tags and for multiple forms to have the same tag.

The third major type of relation is *one-to-many*. This is appropriate

for situations where an object has an attribute whose value is a collection of zero or more objects yet no other object can have those objects. In the OLD, forms and their translations are in a one-to-many relation since a sentence, say, should be able to have multiple translations yet allowing multiple sentential forms to have the same translation is undesirable (for reasons discussed below). A one-to-many relation requires no third relational table; the table on the many side simply references that on the one side, e.g., translations have columns for *form identifier* which contain values that reference unique form rows.

Relational database management systems (RDBMSs) such as MySQL and SQLite (the ones usable by an OLD application) are essentially interfaces for interacting with a relational data structure. The RDBMS interprets structured query language (SQL) commands that allow the data to be read, created, updated, destroyed and searched.

The OLD makes use of the Python module SQLAlchemy and, in particular, its object-relational mapper (ORM) which transparently maps objects to the appropriate rows and relations in the underlying relational database. This tool makes it easier to extract the conceptually simpler objects from the multi-table rows and relations of the underlying database.

A.2 OLD objects

This section catalogues the objects encoded within the data structure of the OLD. It is organized alphabetically by object and each object's attributes are listed alphabetically.

Note that this list is out of date in that it is missing descriptions of the following objects: corpus, corpus backup, morpheme language model, morpheme language model backup, morphological parser, morphological parser backup, morphology, morphology backup, phonology, and phonology backup. Readers interested in a deeper understanding of these OLD objects are invited to consult the Python modules for the relevant objects in the source code of the OLD. These modules can be found at <https://github.com/jrwdunham/old/blob/master/onlinelinguisticdatabase/model>.

A.2.1 Application settings

Application settings objects store system-wide settings that affect such things as input validation, morpheme delimiter characters, valid grammaticality values, the name of the language under study, etc.

Two things that should also be mentioned here are that (1) only administrators can modify or create application settings objects and (2) the most recently created application settings object is considered the currently active one, i.e., the one that determines the settings for the entire application.

A.2.1.1 Broad phonetic inventory

The value of the broad phonetic inventory attribute is a comma-delimited string representing the inventory of graphemes (i.e., single characters or strings of characters) that should be used to construct values for the phonetic transcription attribute of form objects. The space character should not be included as a grapheme since the validation functionality will allow it by default.

A.2.1.2 Broad phonetic validation

The broad phonetic validation attribute determines how or whether the input to the phonetic transcription attribute of forms is validated. The permissible values of the broad phonetic validation attribute “Error”, “Warning” and “None”. If the value is “Error”, then the OLD will not permit a form to be created or updated if its phonetic transcription value cannot be constructed using the graphemes in the broad phonetic inventory plus the space character.

A.2.1.3 Grammaticalities

The grammaticalities attribute holds a comma-delimited list of grammaticality values that will be the available options for the grammaticality attributes of form objects and the grammaticality attributes of translation objects. The default value for this attribute is “*,#,?”.

A.2.1.4 Input orthography

The input orthography is a reference to an existing orthography object. An orthography is essentially a list of graphemes (like an inventory) but with some extra settings (cf. § A.2.10). The purpose of a system-wide input orthography is to allow users to enter form transcriptions (and possibly also morpheme breaks) using one orthography (i.e., their input orthography) and have these transcriptions translated into another orthography (i.e., the storage orthography) for storage in the database. When outputting the forms, the system would then re-translate them from the storage orthography into

the output orthography. Previous OLD applications implemented this orthography conversion server-side. However, with the new architecture of the OLD ≥ 1.0 this added complication seems best implemented client-side as user-specific orthography conversion. Therefore, the input orthography attribute of the application settings object may be removed in future versions of the OLD.

A.2.1.5 Metalanguage identifier

The value of the metalanguage identifier attribute is a three-character string from the ISO 639-3 standard which unambiguously identifies the metalanguage of the application, i.e., the language used in the analysis and documentation of the object language. OLD applications store the ISO 639-3 data as language objects that can be requested like any other object. The default value for the metalanguage identifier attribute is “eng”.

A.2.1.6 Metalanguage inventory

The value of the metalanguage inventory attribute is a comma-delimited string representing the inventory of graphemes that should be used to construct the translation transcriptions of form objects. Note that the OLD is not set up to use this inventory for validation.

A.2.1.7 Metalanguage name

The value of the metalanguage name is the name of the language that is used in the analysis (and translation) of the language under study (the object language). The default value for this attribute is “English”.

A.2.1.8 Morpheme break is orthographic

The value of the morpheme break is orthographic attribute controls what characters the system will expect to find in the values of the morpheme break attribute of forms. If morpheme break is orthographic is set to “true” (or “yes”, “on” or “1”), then the system will expect the morpheme break value to be constructed using the graphemes defined in the storage orthography attribute; if it is set to “false” (or “no”, “off” or “0”), the system will expect graphemes from the phonemic inventory in the value of this attribute.

A.2.1.9 Morpheme break validation

The morpheme break validation attribute determines how or whether the input to the morpheme break attribute of forms is validated. The permissible values of the morpheme break validation attribute are “Error”, “Warning” and “None”. If the value is “Error”, then the OLD will not permit a form to be created or updated if its morpheme break value cannot be constructed using the graphemes of the relevant orthography/inventory (cf. the morpheme break is orthographic attribute) plus the space character.

A.2.1.10 Morpheme delimiters

The morpheme delimiters attribute holds a comma-delimited list of characters that the system should expect users will employ when segmenting morpheme transcriptions or morpheme glosses in the morpheme break and morpheme gloss fields, respectively. The default value for this attribute is “-,=”. If morpheme break validation is enabled, then these delimiter char-

acters will be permitted in the morpheme break values in addition to the graphemes of the specified orthography/inventory.

A.2.1.11 Narrow phonetic inventory

The value of the narrow phonetic inventory attribute is a comma-delimited string representing the inventory of graphemes (i.e., single characters or strings of characters) that should be used to construct narrow phonetic transcriptions, i.e., to construct values for the narrow phonetic transcription attribute of form objects. The space character should not be included as a grapheme since the validation functionality will allow it by default.

A.2.1.12 Narrow phonetic validation

The narrow phonetic validation attribute determines how or whether the input to the narrow phonetic transcription attribute of forms is validated. The permissible values of the narrow phonetic validation attribute are “Error”, “Warning” and “None”. If the value is “Error”, then the OLD will not permit a form to be created or updated if its narrow phonetic transcription value cannot be constructed using the graphemes in the narrow phonetic inventory plus the space character.

A.2.1.13 Object language identifier

The value of the object language identifier attribute is a three-character string from the ISO 639-3 standard which unambiguously identifies the language being documented using the application, i.e., the object language.

OLD applications store the ISO 639-3 data as language objects that can be requested like any other object.

A.2.1.14 Object language name

The value of the object language name is the name of the language that is being documented and analyzed using the OLD web service. This name should correspond with the ISO 639-3 object language identifier and *may* be the value of the ISO 639-3 `Ref_Name` attribute. However, sometimes the “reference name” for the language provided by the standard is not that which the fieldworkers or community prefer to use and there should be no sense of obligation to use it.

A.2.1.15 Orthographic validation

The orthographic validation attribute determines how or whether the input to the transcription attribute of forms is validated. The permissible values of the orthographic validation attribute are “Error”, “Warning” and “None”. If the value is “Error”, then the OLD will not permit a form to be created or updated if its transcription value cannot be constructed using the graphemes in the storage orthography plus the space character and the specified punctuation.

A.2.1.16 Output orthography

The output orthography is a reference to an existing orthography object object. An orthography is essentially a list of graphemes (like an inventory) but with some extra settings (cf. § A.2.10). The purpose of a system-wide output

orthography is to allow for the possibility that users will enter form transcriptions (and possibly also morpheme breaks) using one orthography (i.e., the input orthography) but that these transcriptions will be translated into another orthography (i.e., the storage orthography) for storage in the database. When outputting the forms, the system would then re-translate them from the storage orthography into the output orthography. Previous OLD applications implemented this orthography conversion server-side. However, with the new architecture of the OLD ≥ 1.0 this added complication seems best implemented client-side as user-specific orthography conversion. Therefore, the output orthography attribute of the application settings object may be removed in future versions of the OLD.

A.2.1.17 Phonemic inventory

The value of the phonemic inventory attribute is a comma-delimited string representing the inventory of phonemes that should be used to construct morpheme segmentations in the morpheme break attribute of form resources.

A.2.1.18 Punctuation

The punctuation attribute holds a string representing a list of punctuation characters. There is no delimiter: each character in the string is considered a punctuation character. Thus the default value of this field is given in (58).

(58) . , ; : ! ? ' " ' ‘ ’ “ ” [] { } () -

This results in the following characters being identified as valid punctuation: FULL STOP, COMMA, SEMICOLON, COLON, EXCLAMATION

MARK, QUESTION MARK, APOSTROPHE, QUOTATION MARK, LEFT SINGLE QUOTATION MARK, RIGHT SINGLE QUOTATION MARK, LEFT DOUBLE QUOTATION MARK, RIGHT DOUBLE QUOTATION MARK, LEFT SQUARE BRACKET, RIGHT SQUARE BRACKET, LEFT CURLY BRACKET, RIGHT CURLY BRACKET, LEFT PARENTHESIS, RIGHT PARENTHESIS, HYPHEN-MINUS. When orthographic validation is enabled, the system will allow the punctuation characters specified in this attribute to occur in the values of the transcription attribute of forms.

A.2.1.19 Storage orthography

The storage orthography is a reference to an existing orthography object. An orthography is essentially a list of graphemes (like an inventory) but with some extra settings (cf. § A.2.10). The storage orthography defines the character sequences that should be used to create form transcription values. If the morpheme-break-is-orthographic attribute is set to “true”, then the form morpheme break values should also be constructed out of the graphemes defined in the storage orthography (plus the morpheme delimiters specified in morpheme delimiters).

The system-wide storage orthography is also a component in an orthography conversion feature. Orthography conversion allows for the possibility that users will enter form transcriptions (and possibly also morpheme breaks) using one orthography (i.e., the input orthography) but that these transcriptions will be translated into another orthography (i.e., the storage orthography) for storage in the database. When outputting the forms, the system would then re-translate them from the storage orthography into

the output orthography. This is useful because it allows contributors to use an orthography that they are comfortable with while still ensuring that all orthographic transcription data are stored in the same orthography underlyingly. Previous OLD applications implemented this orthography conversion server-side. However, with the new architecture of the OLD ≥ 1.0 this added complication seems best implemented client-side as user-specific orthography conversion.

A.2.1.20 Unrestricted users

The `unrestricted users` attribute is a collection of user objects which identifies the set of users that are to be identified as *unrestricted*. Such users are authorized to access restricted form, file and collection resources while contributors and viewers who are not unrestricted (i.e., who are *restricted*) are unable to view (or, *a fortiori*, update) such resources.

A.2.2 Collection

OLD collection models are documents that can contain both text (with markup) and references to form models in the values of their contents attributes. They can be used for a number of purposes: to create a simple list of forms, to write an academic paper or a lesson plan, to document a conversation or narrative, etc. The value of the `contents` attribute is a document written using one of the lightweight markup languages `reStructuredText` or `Markdown`. OLD collections can embed other OLD collections via reference. As `reStructuredText` or `MarkDown` documents, they can be converted to HTML and, in the case of collections written using `reStructuredText`, they

can be converted to (Xe)LaTeX (whence to PDF) and Open Document Format (i.e., .odt; whence to Word, i.e., .doc).

A.2.2.1 Contents

The value of the contents attribute is a string that constitutes the content of the collection. If markup is used, it should be the markup specified in the markup language attribute.

The value of this attribute can contain references to form models in the database. These references are strings like `form[136]` or `Form[136]`, i.e., the string “form” or “Form”, followed by a left bracket “[”, followed by a valid form model id, followed by a right bracket “]”. The reference “form[136]” would result in the form with identifier 136 being associated to the collection, i.e., the value of the collection’s forms attribute would contain that form.

Note that the value of the contents attribute need not contain any markup or other text. That is, it may simply be a string consisting of references to forms.

Here is an example of a well-formed contents value that uses the Markdown markup language and contains a reference to the form with identifier 136:

```
Chapter 2
=====
```

```
Section containing a list
-----
```

- * Item 1
- * Item 2

Section containing forms

form[136]

It is also possible to reference another collection within the value of the contents attribute. This causes the contents of the first collection to behave as though it contained the contents of the referenced collection in its contents value at the point of reference. For example, consider collection *C2* below which references collection *C1* (with id 3) from above.

Chapter 1
=====

Section containing prose

This section argues for the claim that ...

Section containing forms

form[135]

collection[3]

When collection *C2* is created, the system will generate the following value for *C2*'s contents unpacked attribute:

Chapter 1
=====

Section containing prose

This section argues for the claim that ...

Section containing forms

form[135]

Chapter 2
=====

Section containing a list

- * Item 1
- * Item 2

Section containing forms

form[136]

The above contents unpacked value will be used to extract the form references of the collection and to generate the value of the html attribute. That is, collection *C2* will be associated to forms 135 and 136. Note that collection-collection references can be nested, i.e., collections can reference collections which reference other collections, etc.

A.2.2.2 Contents unpacked

The value of the contents unpacked attribute is the value of the contents attribute when all of its collection references are replaced with the contents of the collections referred to. These referred-to collections can refer to others in turn and all such references are replaced by the appropriate contents values. The form objects associated to a collection are calculated by gathering all

of the form references in the value of the contents unpacked attribute.

A result of collection-to-collection referencing is that the contents and forms values of a collection may be altered by updates to other collections. The system transparently handles this percolation of collection updates.

A.2.2.3 Date elicited

The date elicited attribute is a user-supplied date value which indicates the date when the collection was elicited. The date must be provided in mm/dd/yyyy format. This is applicable to collections that represent records of events, e.g., elicitation sessions, recordings of stories, etc.

A.2.2.4 Date-time entered

The value of the date-time entered attribute is a UTC timestamp generated by the system when a collection is created. Note that this value is distinct from the date-time modified attribute that is common to all model types since that value is generated upon both creation *and* update requests while the date-time entered value is only generated upon creation requests and is not altered thereafter.

A.2.2.5 Description

The value of the description attribute is a user-supplied string that describes the collection.

A.2.2.6 Elicitor

The elicitor attribute references a valid user model who is the elicitor of the collection.²⁷⁰ This attribute may not be appropriate for all collection types.

A.2.2.7 Enterer

The enterer attribute references the user model whose account was used to create the collection. This value is generated automatically by the system upon collection creation.

A.2.2.8 Files

A collection may be associated to zero or more files objects via the files attribute which references a collection²⁷¹ of file objects. Files are OLD objects that represent a digital file (e.g., an audio, video or image file) along with metadata. An example use case would be a collection that represents an elicitation session and which is associated to one or more files whose file data are large audio recordings of the session. See § A.2.5 for details on the structure of file objects.

²⁷⁰As things stand, an elicitor must always be a registered OLD user. This is problematic for cases where a user is entering data from someone who is not a registered user and for whom it does not make sense to create a user account (e.g., in the case of a deceased fieldworker). To handle this type of case, the OLD should be modified to include a more general type of *person* object which could be subclassed to form user and speaker objects.

²⁷¹Note the distinction between OLD *collections* which are a type of model and *collections* as a type of object attribute which references a set of zero or more other objects. For example, the value of the files attribute of form objects is a collection of file models and is an example of a collection in the second sense.

A.2.2.9 Forms

A collection may be associated to zero or more forms. These are stored in the forms attribute, which references a collection of form objects. Forms are associated to a collection by being referenced in the value of the contents attribute of the collection (cf. § A.2.2.1).

A.2.2.10 HTML

The value of the HTML attribute is a string of HTML that is generated by the system using the value of the contents unpacked attribute and the markup-to-HTML function corresponding to the markup language specified in the markup language attribute. Note that while the HTML could be generated in the user-facing application, there is not, to my knowledge, a JavaScript implementation of the reStructuredText-to-HTML algorithm; therefore the HTML generation is performed server-side. Note also that form references are left as-is, which is to say that no HTML representation of the form data is generated. This is left as a task for the user-facing application since applications will have their own method(s) of displaying forms.

A.2.2.11 Markup language

The value of the markup language attribute is one of “Markdown” or “reStructuredText”. Markdown and reStructuredText are *lightweight markup languages*, i.e., systems for annotating a document that are designed to be easy to read in raw form. If no value is specified, “reStructuredText” is considered the default.

A.2.2.12 Source

The source attribute references a valid source model that indicates the textual (or other) source of the collection. This is useful for when the content of a collection is taken from another document and attribution is necessary. The structure of the source model is based on the BibTeX format. See § A.2.12 for details.

A.2.2.13 Speaker

The speaker attribute references a valid speaker model who is the speaker or consultant of the collection. As with attributes like elicitor, the speaker attribute may not be appropriate for all collection types.

A.2.2.14 Tags

A collection may be associated to zero or more tags and these associations are stored in the tags attribute. Tags are user-defined objects that can be used to arbitrarily categorize other OLD objects. If a collection is to be restricted, the special “restricted” tag should be associated to it. See § A.2.15 for details.

A.2.2.15 Title

The value of the title attribute is a string that is the title of the collection. All collections must have a title and no title may exceed 255 characters.

A.2.2.16 Type

The value of the type attribute is used to classify the collection and may affect how it is displayed or exported. The permitted values are “story”, “elicitation”, “paper”, “discourse” and “other”.

A.2.2.17 URL

The value of the URL attribute is not actually a valid URL but something more akin to the *path* component of a URL. That is, it is a string composed of any of the 26 letters of the English alphabet (including uppercase versions), the underscore “_”, the forward slash “/” and the hyphen “-”. The URL value must not exceed 255 characters. At present the OLD qua web service does not make use of this attribute. However, it may be used by a user-facing application to allow users to navigate to a specific collection using something more meaningful than an integer identifier. For example, on a web application front-end to an OLD web service with the URL `http://www.xyz-old.org`, one might navigate to a representation of the collection entitled “Magnum Opus” by entering `http://www.xyz-old.org/magnum_opus` in the address bar (where “magnum_opus” is the value of the URL attribute.)

A.2.2.18 UUID

The value of the UUID attribute is a universally unique identifier (UUID), i.e., a number represented by 32 hexadecimal digits displayed in five groups using four hyphens. A valid UUID is a 36-character string that looks like

aba3ea8d-b56f-4934-a8f7-68cba500f411. The system generates a random (and likely unique²⁷²) UUID value for each newly created collection object. These values are used to associate collection backups to the collections whose past data they store.

A.2.3 Collection backup

A collection backup object is created whenever and only whenever a collection is updated or deleted. The collection backup object receives all of the attributes of the object that it backs up. It also has some additional attributes, viz. collection identifier and backuper. The value of the collection identifier attribute is the value of the identifier attribute of the collection that was backed up to create the present collection backup model. The value of the backuper attribute is a JSON object representing the user who created the backup (by deleting or updating the collection). In general, the values of the relational attributes of the collection (i.e., the attributes that refer to other models) are serialized as JSON objects in the collection backup object. For example, the value of the speaker attribute is such a JSON object and the value of the files attribute is a JSON array of such objects representing file objects. Since form models have many attributes and since collection models will, typically, be associated to many form models, the

²⁷²The chances of randomly generating two identical UUIDs is extremely remote, cf. <https://en.wikipedia.org/wiki/UUID>. If two OLD collections should happen to have the same UUID, then their backup copies will have the same UUID values also. However, collection objects also have integer id values which should be unique. The only situation in which two objects will have the same id value is where one is deleted and the RDBMS reuses the deleted object's id for a newly created one (something which is done only by certain RDBMSs). This makes it even more unlikely that the histories of two objects could become mixed up because of identical UUID and id values.

forms attribute of a collection backup model is simply a JSON array of form identifier values. If the collection has just been deleted, then the value of the date-time modified value of the collection backup will be the UTC date-time at the time of deletion.

A.2.4 Elicitation method

Elicitation method objects represent a set of tags for categorizing the way in which a form was elicited. For example, sometimes a researcher asks a consultant “How do you say ‘Every man loves a woman.’?” An elicitation method used to categorize forms elicited in this way might have a name value of “translated English”. Sometimes a researcher asks a consultant “Does this sound like a good sentence: ‘Il y a une femme que tous les hommes aiment.’?” The elicitation method for such forms might have a name of “judged object language utterance of researcher”.

A.2.4.1 Description

The value of the description attribute is a user-supplied string that describes the elicitation method and (perhaps) provides guidance on its use.

A.2.4.2 Name

The value of the name attribute is an obligatory, user-supplied string of no more than 255 characters which must be unique among all other elicitation method names.

A.2.5 File

OLD file objects are digital files with additional attributes. From the language fieldworker's point of view, they are the audio/video recordings of linguistic fieldwork as well as image, audio or video files that may be used to elicit speech or even the documents (such as PDFs of handouts or pedagogical materials) that are in some way related to language data.

There are three types of file object and while each share a common set of core attributes, they have attributes unique to their type as well. *Local* files are stored on the file system of an OLD application's server. *Subinterval-referencing* files get their file content from a local audio/video file (their parent file) and have start and end attributes which reference start and end positions in the parent file. *Externally hosted* files have content stored on another server and have URL attributes for locating that content.

A.2.5.1 Date elicited

The date elicited attribute is a user-supplied date value which indicates the date when the file was elicited, if applicable, e.g., when an audio recording of an elicitation was made. The date must be provided in mm/dd/yyyy format.

A.2.5.2 Date-time entered

The value of the date-time entered attribute is a UTC timestamp generated by the system when a file is created. Note that this value is distinct from the date-time modified attribute that is common to all model types since that value is generated upon creation *and* update requests while the date-time

entered value is only generated upon creation requests and is not altered thereafter.

A.2.5.3 Description

The value of the description attribute is a user-supplied string that describes the file.

A.2.5.4 Elicitor

The elicitor attribute references a valid user model who is the elicitor of the file, if applicable.

A.2.5.5 End

The value of the end attribute is a number (integer or float) representing the end of the subinterval in seconds of a subinterval-referencing file. For example, consider the subinterval-referencing file *F2* which references the audio file *F1* as its parent file. A value of 3.7 for the end attribute of *F2* means that the audio of *F2* is a sub-portion of the audio of *F1* which ends at 3.7 seconds. Note that only subinterval-referencing files should have values for the end attribute.

A.2.5.6 Enterer

The enterer attribute references the user object whose account was used to create the file. This value is generated automatically by the system upon file creation.

A.2.5.7 File name

The file name attribute holds the name of the file as it is stored in the file system. When a local file is created, a non-empty file name value must be provided in the input parameters. While Unicode (i.e., non-ASCII) characters are permitted in the filename value, the system removes certain characters (QUOTATION MARK (”), APOSTROPHE (‘), the path separator (/ on Unix systems) and the null byte) and replaces spaces with underscores. If a file with the resulting name already exists in the directory that holds local file data, then the system will alter the name (by inserting an underscore followed by a string of eight random characters between the end of the file name and its extension) until a unique one is found. The resulting string becomes the value of the file name attribute. So, for example, if a file create request contains “john’s file.wav” as the value of the file name parameter and if `johns_file.wav` already exists in the OLD application, then the file data will be saved to something like `johns_file_3Df6Nop0.wav` and the value of the file name attribute of the file model will be “johns_file_3Df6Nop0.wav”.

A.2.5.8 Forms

A file object may be associated to zero or more forms. On file create and update requests, associated forms are specified by providing an array of valid form identifiers as the value of the forms attribute. When JSON object representations of file models are returned, the value of the forms attribute is an array of JSON objects representing the associated forms.

A.2.5.9 Lossy file name

If the OLD is configured to create reduced-size copies of uploaded files and if the requisite dependencies are installed (i.e., PIL or FFmpeg), then the system will create reduced-size (i.e., lossy) copies of the files in a separate directory and the lossy file name attribute will return the name of the reduced-size copy in that directory. This is useful for when contributors upload lossless audio files like WAV files. To facilitate quicker response time, a lossy (i.e., .mp3 or .ogg) copy may then be streamed across the network instead of the original lossless version of the file.

A.2.5.10 MIME type

MIME types, also known as Internet Media Types, are standardized strings used to categorize types of binary files. An OLD web service will ascertain the MIME type of an uploaded file based on its contents. If the MIME type is in the list of those allowed, then the value of the MIME type attribute will be assigned to the ascertained MIME type string. The types of files that can be uploaded to an OLD web service are Portable Document Format; GIF image; JPEG JFIF image; Portable Network Graphics; MP3 or other MPEG audio; Ogg Vorbis, Speex, Flac and other audio; WAV audio; MPEG-1 video with multiplexed audio; MP4 video; Ogg Theora or other video (with audio); QuickTime video; and Windows Media Video.

A.2.5.11 Name

Externally hosted and subinterval-referencing files may supply a value for the name attribute. Since these types of files do not have values for the file name attribute, the name attribute can be useful in identifying them. For local files the system automatically sets the name attribute to the value of the file name attribute. If a subinterval-referencing file creation request does not include a non-empty name value, then the value assigned to that attribute is the value of the file name attribute of the subinterval-referencing file's parent file.

A.2.5.12 Parent file

Subinterval-referencing files are identified by possession of a non-empty parent file attribute. The value of this attribute is a reference to an existing local file. The parent file must be an audio or video file. The subinterval-referencing file gets its file data from its parent file.

A.2.5.13 Password

The password attribute can be specified for externally hosted file models that require a password in order for the external host to serve the file. Note that this value will be available to all users of the system and should *not* therefore be a password used for other purposes, e.g., to log in to the OLD web service itself.

A.2.5.14 Size

Local file models have a value for the size attribute which is an integer representing the size of the binary file in bytes. This value is generated by the system upon a successful file creation request.

A.2.5.15 Speaker

The speaker attribute references a valid speaker object who is the speaker or consultant of the file. This is appropriate in cases where the file is, say, an audio recording of a speaker telling a story or a recording of an elicitation session with a particular consultant.

A.2.5.16 Start

The value of the start attribute is a number (integer or float) representing in seconds the beginning of the subinterval of a subinterval-referencing file. For example, consider the subinterval-referencing file *F2* which references the audio file *F1* as its parent file. A value of 2.1 for the start attribute of *F2* means that the audio for *F2* is the sub-portion of *F1*'s audio beginning at 2.1 seconds and continuing until the specified end position. Note that only subinterval-referencing files should have values for the start attribute.

A.2.5.17 Tags

A file may be associated to zero or more tags. Tags are user-defined objects that can be used to arbitrarily categorize other OLD models. If a file is to be restricted, then the special “restricted” tag should be associated to it.

See the § A.2.15 for more details on the tag model.

A.2.5.18 URL

Externally hosted files are identified by possession of a non-empty value for the URL attribute. The value should be a valid URL that will serve the content of the file when requested. This value will allow user-facing applications to display (i.e., embed) the file content of externally hosted file models.

A.2.5.19 Utterance type

Files that represent recordings of utterances should be categorized using the utterance type attribute. Valid values are “None”, “Object Language Utterance”, “Metalanguage Utterance” and “Mixed Utterance”.

Here is a potential use case scenario for this attribute. Consider an OLD web service that is being used to study the Blackfoot language and imagine a file model $F1$ whose binary data is a WAV file audio recording of a speaker saying “oki”, which means “hello” in Blackfoot. Now imagine a second file, $F2$ whose binary data is another WAV file recording of the speaker saying “hello”. Assume that the utterance type value of $F1$ is “Object Language Utterance” (since it is a recording of an utterance of the object language, i.e., Blackfoot) and assume that the utterance type value of $F2$ is “Metalanguage Utterance” (since it is a recording of an utterance in the language of analysis and translation, i.e., English). Now imagine a form F whose transcription is “oki” and whose only translation is “hello” and which is associated to files $F1$ and $F2$. If there are a good number of forms like F , then an application

making use of this OLD web service would be able to assume that $F1$, being an object language utterance associated to F is a recording of a speaker uttering the linguistic form that is transcribed in F . Such an application could then use such forms to automatically generate audio/textual language learning games or talking dictionaries.

A.2.6 Form

An OLD form model represents a linguistic form in a very general sense; that is, it can represent a lexical item abstracted from any elicitation or recording event as well as a word, phrase or sentence uttered on a particular occasion by a particular speaker.

A.2.6.1 Break-gloss-category

The break-gloss-category attribute stores a system-generated string which merges the values of the morpheme break, morpheme gloss and syntactic category string attributes. For example, the break-gloss-category value of a form with “chien-s” as its morpheme break, “dog-PL” as its morpheme gloss string and “N-Num” as its syntactic category would be “chien|dog|N-s|PL|Num”. Since the break-gloss-category value is searchable, it can be used to filter forms according to presence/absence of a specific morpheme.

A.2.6.2 Collections

A form may be associated to zero or more collections. Collections are documents that typically reference, and are associated to, multiple forms. Note

that such associations are *not* created during form creation or updating but during collection creation. See § A.2.2 for details.

A.2.6.3 Comments

The comments attribute is an open-ended field that may contain any comments about the form or any data that do not fit neatly into the standard attributes of the form resource. If multiple forms are to be tagged or classified in some way, it is better to use the tags attribute for this purpose and not the comments attribute.

A.2.6.4 Date elicited

The date elicited attribute is a user-supplied date value which indicates the date when the form was elicited. The date must be provided in mm/dd/yyyy format. For abstract lexical forms this value may not be appropriate.

A.2.6.5 Date-time entered

The value of the date-time entered attribute is a UTC timestamp generated by the system when a form is created. Note that this value is distinct from the date-time modified attribute that is common to all model types since that value is generated upon creation *and* update requests while the date-time entered value is only generated upon creation requests and is not altered thereafter.

A.2.6.6 Elicitation method

The elicitation method attribute references a valid elicitation method object that classifies the way in which the form was elicited. See § A.2.4 for details.

A.2.6.7 Elicitor

The elicitor attribute references a valid user model who is the elicitor of the form.

A.2.6.8 Enterer

The enterer attribute references the user object whose account was used to enter the form. This value is generated automatically by the system upon form creation.

A.2.6.9 Files

A form may be associated to zero or more files via the files attribute which references a collection of file objects. Files are OLD objects that represent a digital file (e.g., an audio, video or image file) supplemented with additional attributes, e.g., a description or the size of the file. See § A.2.5 for details on the structure of file objects.

A.2.6.10 Grammaticality

The grammaticality attribute stores the grammaticality value assigned to the form. This is a forced-choice attribute whose options are defined by the administrator(s) of the system in the grammaticalities attribute of the active

application settings resource. Typically, the available grammaticalities will be a list such as “*”, “?”, “#”, “**”, etc.

A.2.6.11 Memorizers

The memorizers attribute holds a collection of zero or more user models corresponding to the users who have memorized, or remembered, this form.

A.2.6.12 Morpheme break

The morpheme break attribute holds a representation of the morphological analysis of a linguistic form, i.e., a segmentation of a (presumably) phonemic transcription into component morphemes. The maximum length is 255 characters. The system will expect words to be split by whitespace and morphemes by the delimiters specified in the morpheme delimiters attribute of the active application settings. By specifying appropriate values for the morpheme break validation, morpheme-break-is-orthographic and phonemic-inventory or storage-orthography attributes of the active application settings resource, it is possible to ensure that data input to this attribute are validated against the specified orthography/inventory and delimiters.

A.2.6.13 Morpheme break identifiers

The value of the morpheme break identifiers attribute is a system-generated JSON array that contains references to all matches found for each morpheme listed in the morpheme break attribute. The system generates this value every time a form is created or updated. This facilitates morphological

analysis consistency visualization while avoiding costly, repetitive database queries each time a form is retrieved.

A.2.6.14 Morpheme gloss

The morpheme gloss attribute holds a string of glosses (separated by spaces and morpheme delimiters) where each such gloss should correspond to a morpheme whose phonemic shape is represented in the morpheme break field. The maximum length is 255 characters. As with the morpheme break field, the gloss “words” in this field should be delimited using whitespace and the glosses within words should be delimited using the morpheme delimiters specified in the active application settings object.

A.2.6.15 Morpheme gloss identifiers

The value of the morpheme gloss identifiers attribute is a system-generated JSON array that contains references to all matches found for each morpheme gloss listed in the morpheme gloss attribute. The system generates this value every time a form is created or updated (cp. the A.2.6.13 paragraph above).

A.2.6.16 Narrow phonetic transcription

The narrow phonetic transcription attribute holds a narrow phonetic transcription of the linguistic form. The maximum length is 255 characters. By specifying a value for the narrow phonetic inventory attribute of the active application settings and setting that same resource’s narrow phonetic validation attribute to “Error”, it is possible to configure narrow phonetic

transcription validation so that values not generable using the specified inventory are rejected.

A.2.6.17 Phonetic transcription

The phonetic transcription attribute holds a phonetic transcription of the linguistic form. By convention, this is a *broad* phonetic transcription. The maximum length is 255 characters. By specifying a value for the broad phonetic inventory attribute of the active application settings and setting that same resource's broad phonetic validation attribute to "Error", it is possible to configure phonetic transcription validation so that values not generable using the specified inventory are rejected.

A.2.6.18 Semantics

The value of the semantics attribute is canonically a semantic representation of the form, e.g., a denotation. The maximum length is 1023 characters. At some future point candidate values for this attribute may be auto-generated.

A.2.6.19 Source

The source attribute references a valid source object that indicates the textual (or other) source of the form. This is useful for when data are taken from papers or dictionaries and need to be attributed. The source object is based on the BibTeX format. See § A.2.12 for details.

A.2.6.20 Speaker

The speaker attribute references a valid speaker object who is the speaker or consultant of the form.

A.2.6.21 Speaker comments

The speaker comments attribute holds comments made about the form by the speaker or consultant.

A.2.6.22 Status

The status attribute encodes the status of the form with respect to its verification. At present, the two licit values are “tested” and “requires testing”. Usage of this attribute permits researchers to enter forms not yet tested (i.e., verified with a language consultant) in order to prepare for a planned elicitation session.²⁷³

A.2.6.23 Syntactic category

The syntactic category attribute references a valid syntactic category object that categorizes the form. For example, a form like “chien” might have a syntactic category value which references a syntactic category object whose name attribute is “N”. See § A.2.14 for details.

²⁷³The OLD 0.2 interface does not currently make use of the status attribute on forms. When an interface is developed which does make use of it, it will be important to ensure that unreal data, i.e., that which requires testing, be clearly represented as such and not be returned in search results without an explicit request to do so by the user.

A.2.6.24 Syntactic category string

The syntactic category string attribute holds a system-generated value which is a string of syntactic category names corresponding to the morphemes specified by the creator/updater of the form. That is, the system inspects the values of the morpheme break and morpheme gloss fields and searches the database for matches to the specified morpheme/gloss pairs; the names of the syntactic categories of the matches are used to generate the value for the syntactic category string attribute. By searching forms based on patterns in this field it is possible to filter the database according to higher-level morphological or syntactic patterns.

A.2.6.25 Syntax

The value of the syntax attribute is a syntactic representation of the form, e.g., a phrase structure tree in bracket notation. The maximum length is 1023 characters. At some future point candidate values for this attribute may be auto-generated.

A.2.6.26 Tags

A form may be associated to zero or more tags. Tags are user-defined objects that can be used to arbitrarily categorize other OLD models. An example usage would be to define a tag model with a name value of “VP ellipsis” and use that tag to categorize forms that exhibit the phenomenon. If a form is to be restricted, then the special “restricted” tag should be associated to it; similarly, if the form documents a foreign word, then it should be associated

to the special “foreign word” tag.

A.2.6.27 Translations

A form object must have at least one translation but may have more. The translations of a form are each translation objects that are listed in the translations attribute of the form. (In the relational database schema, the form and translation tables are in a one-to-many relationship.) Forms with multiple translations, e.g., sentences with multiple valid translations, should use separate translation models for each such translation. Translation models can also have grammaticalities (cf. the grammaticality attribute). This feature may be used to indicate a translation that is not appropriate to a grammatical form. Thus, as a simplistic example, “chien” may be translated as “dog” and “*wolf” using two translation models.

A.2.6.28 Transcription

The transcription attribute holds transcriptions of linguistic forms. By convention, these are expected to be written in an orthography of the object language. The maximum length is 255 characters. Every form must have a transcription value. It is possible to specify a storage orthography in the active application settings resource and configure form transcription validation so that values not generable using the orthography are rejected.

A.2.6.29 UUID

The value of the UUID attribute is a universally unique identifier (UUID), i.e., a number represented by 32 hexadecimal digits displayed in five groups

using four hyphens. A valid UUID is a 36-character string that looks like `aba3ea8d-b56f-4934-a8f7-68cba500f411`. The system generates a random UUID value for each newly created form object. These values are used to associate form backups to the forms whose past data they store.

A.2.6.30 Verifier

The verifier attribute references a valid user object representing the field-worker who has verified the form. This is useful, for example, in a case where one researcher finds that a form they have elicited has already been stored in the database and they do not want to record a duplicate entry. Oftentimes, however, it is desirable to enter a duplicate entry.

A.2.7 Form backup

A form backup object is created whenever and only whenever a form object is updated or deleted. The form backup object receives all of the attributes of the form whose past values it stores. It also has some additional attributes, viz. form identifier and backuper. The value of the form identifier attribute is the value of the identifier attribute of the form that was backed up to create the present form backup object. The value of the backuper attribute is a JSON object representing the user who created the backup (by deleting or updating the form). In general, the values of the relational attributes of the form (i.e., the attributes that refer to other objects) are converted to JSON object representations in the form backup object. For example, the value of the speaker attribute is such a JSON object and the value of the files attribute is a JSON array of such objects representing file objects. If

the form has just been deleted, then the value of the date-time modified value of the form backup will be the UTC date-time at which the backup occurred.

A.2.8 Form search

The form search object stores searches over the set of forms so that these searches can be saved for later use and shared with other users of the system.

A.2.8.1 Description

The value of the description attribute is a user-supplied string that describes the search object.

A.2.8.2 Name

The value of the name attribute is a user-supplied string used to identify the search resource. Names are obligatory, may not exceed 255 characters, and no two searches may have the same name.

A.2.8.3 Search

The value of the search attribute is a JSON object representing the search. If the user-supplied search object is not well-formed, the system will prevent the form search resource from being created or updated. The search object has an obligatory filter attribute and an optional order by attribute (see below). The values of both of these attributes are arrays. The former contains a specification of the query, i.e., a structured coordination of filter

expressions. The latter contains a specification of the order in which the results should be returned.

A.2.8.4 Searcher

The searcher attribute references the user object whose account was used to create the form search. This value is generated automatically by the system upon form search creation.

A.2.9 Language

Each language object represents a language listed in the ISO 639-3 standard.²⁷⁴ These objects are present in all OLD application databases. Existing language objects cannot be updated and new ones cannot be created. The purpose of this set of objects is to provide options for the metalanguage and object language identifier and name attributes of application settings objects.

The language objects are unique among OLD models in that the values of their identifier attributes are not integers but the unique three-character strings specified in ISO 639-3. Another thing to note is that the value of the reference name attribute of the language has no special importance and OLD administrators are encouraged to use whatever language name is deemed most appropriate for their application. That said, care should be taken to attempt to identify the correct three-character identifier for the language being documented via an OLD web service so that this information is unambiguous.

²⁷⁴<http://www-01.sil.org/iso639-3/download.asp>

A.2.10 Orthography

An orthography object is a representation of the graphemes used in a particular writing system. The OLD makes use of orthography objects in order to effect input validation on the orthographic transcription and morpheme break attributes of form objects. Previous versions of the OLD implemented orthography conversion functionality server-side, thus allowing users to enter transcriptions in one orthography and have it converted to another (storage) orthography. However, this functionality will now be the responsibility of any user-facing applications that make use of an OLD web service.

A.2.10.1 Initial glottal stops

Initial glottal stops is a Boolean attribute that defaults to true. It controls whether the orthography marks glottal stops at the beginning of words and can be useful for orthography conversion algorithms.

A.2.10.2 Lowercase

Lowercase is a Boolean attribute that defaults to false. It controls whether the orthography uses only lowercase characters and can be useful for orthography conversion algorithms and for reducing the number of graphemes that must be specified in the orthography attribute.

A.2.10.3 Name

The name attribute holds a name for the orthography. This must be unique among orthography names and may not exceed 255 characters. The name

should facilitate identification of the orthography.

A.2.10.4 Orthography

The value of the orthography attribute is a comma-delimited list of strings representing the graphemes of the orthography. A non-empty value for this attribute is required.

Previous versions of the OLD drew significance from the ordering of the graphemes (i.e., for sorting & alphabetization) and also encouraged bracketing of graphemes into equivalence classes for the purpose of sorting (i.e., “a” and “á” would be sorted equivalently if the orthography contained “..., [a, á], ...”). The OLD web service now leaves orthography conversion to the user-facing applications; therefore, additional conventions for orthography specification (such as the significance of ordering and equivalence bracketing) should be detailed in the documentation of those applications.

As described in § A.2.1, orthography models and, in particular, the values of their orthography attributes are used in input transcription validation.

A.2.11 Page

Page objects allow users to create web pages using a specified markup language. Some of the attributes (e.g., heading or name) may be removed or renamed in future versions of the OLD.

A.2.11.1 Content

The content attribute holds a string representing the content of the page written in the specified markup language.

A.2.11.2 Heading

The value of the heading attribute is a user-supplied string, no longer than 255 characters, which could be used as a heading or title for the page.

A.2.11.3 HTML

The value of the HTML attribute is the HTML generated from the user-supplied content value using the markup-to-HTML function corresponding to the specified markup language.

A.2.11.4 Markup language

The value of the markupLanguage attribute is one of “Markdown” or “reStructuredText”. Markdown and reStructuredText are *lightweight markup languages*. A lightweight markup language is a markup language (i.e., a system for annotating a document) that is designed to be easy to read in its raw form. The system will expect the value of the content attribute to contain markup in the specified markup language and will choose a markup-to-HTML function corresponding to that markup language when generating the HTML of the page. If no value is specified, “reStructuredText” is the default.

A.2.11.5 Name

The value of the name attribute is a string used to identify the page. This value may not exceed 255 characters and a non-empty string must be provided as value.

A.2.12 Source

Sources are references to texts that can be cited in the source attribute of form and collection models. The source schema is that of the BibTeX file format. The OLD validates input to source create and update requests in adherence to the BibTeX format. That is, a source of a given type (i.e., a BibTeX entry type) must have values for all of the required attributes of that type. For example, a source with a type value of “article” must have values for its author, title, journal and year attributes.

OLD source models have attributes corresponding to all of the standard BibTeX field names as well as attributes corresponding to some non-standard ones. The full list of source attributes is given below. In general, the source attribute names match their BibTeX field name counterparts exactly. The exceptions to this are the `key`, `keyField`, `type` and `typeField` attributes which correspond to BibTeX `key`, “key” field name, entry type and “type” field name, respectively. See the relevant subsections below for details.

Like all other OLD models, sources have identifier and date-time modified attributes. Source models also have a `file` attribute for referencing an OLD file model.

The descriptions of the BibTeX field names given in the subsections below are taken, with some modifications, from Kopka and Daly (2004). The restrictions on lengths of attribute values are imposed (somewhat arbitrarily) by the OLD and are not part of the BibTeX specifications.

A.2.12.1 Abstract

An abstract of the work. Maximum length is 1000 characters.

A.2.12.2 Address

Usually the address of the publisher or other type of institution. For major publishing houses, it is recommended that this information be omitted entirely. For small publishers, on the other hand, you can help the reader by giving the complete address. Maximum length is 1000 characters.

A.2.12.3 Affiliation

The author's affiliation. Maximum length is 255 characters.

A.2.12.4 Annote

An annotation. It is not used by the standard bibliography styles, but may be used by others that produce an annotated bibliography.

A.2.12.5 Author

The name(s) of the author(s), in the format described in Kopka and Daly (2004). There are two basic formats: (1) *Given Names Surname* and (2) *Surname, Given Names*. For multiple authors, use the formats just specified and separate each such formatted name by the word “and”. Maximum length is 255 characters.

A.2.12.6 Book title (booktitle)

Title of a book, part of which is being cited. See Kopka and Daly (2004) for details on how to type titles. For book entries, use the title field instead. Maximum length is 255 characters.

A.2.12.7 Chapter

A chapter (or section or whatever) number. Maximum length is 255 characters.

A.2.12.8 Contents

A table of contents. Maximum length is 255 characters.²⁷⁵

A.2.12.9 Copyright

Copyright information. Maximum length is 255 characters.

A.2.12.10 Cross-reference (crossref)

The key value of another source to be cross-referenced. Any attribute values that are missing from the source model are inherited from the source cross-referenced via the cross-reference attribute. Maximum length is 1000 characters.

If a valid key value is supplied as the value of the cross-reference attribute, the system will use the attributes of the cross-referenced source

²⁷⁵The 255 character maximum length here is an arbitrary limit and should probably be increased.

when validating the input. That is, a source whose type value is, for example, “inproceedings” would normally fail validation if it lacks a value for its book title attribute; however, if it cross-references another source whose type value is “proceedings” and which has a content-ful book title value, then it will pass validation. If a valid cross-reference value is passed on input, then, on output, the value of cross-reference source will be an object representing the cross-referenced source.

A.2.12.11 Cross-reference source (crossrefSource)

The value of the cross-reference source attribute is either null or the source object that is cross-referenced via the cross-reference attribute. That is, a valid cross-reference value passed on input will cause the system to set the cross-referenced source as the value of the cross-reference source attribute. When returning a JSON representation of the original source, the value of the cross-reference source attribute will be a JSON object representing the cross-referenced source.

A.2.12.12 Edition

The edition of a book—for example, “Second”. This should be an ordinal, and should have the first letter capitalized, as shown here; the standard styles convert to lower case when necessary. Maximum length is 255 characters.

A.2.12.13 Editor

Name(s) of editor(s), typed as indicated in Kopka and Daly (2004). At its most basic, this means either as *Given Names Surname* or *Surname, Given*

Names and using “and” to separate multiple editor names. If there is also a value for the author attribute, then the editor attribute gives the editor of the book or collection in which the reference appears. Maximum length is 255 characters.

A.2.12.14 File

Source models may reference an OLD file object via the file attribute, thus permitting the association to a source of a document containing the source text itself. Note that the file attribute does not correspond to a standard BibTeX field name.

A.2.12.15 How published (howpublished)

How something has been published. The first word should be capitalized. Maximum length is 255 characters.

A.2.12.16 Institution

The sponsoring institution of a technical report. Maximum length is 255 characters.

A.2.12.17 ISBN

The International Standard Book Number. Maximum length is 20 characters.

A.2.12.18 ISSN

The International Standard Serial Number. Used to identify a journal. Maximum length is 20 characters.

A.2.12.19 Journal

A journal name. Abbreviations are provided for many journals. Maximum length is 255 characters.

A.2.12.20 Key

The OLD source key field is the BibTeX key, i.e., the unique string used to unambiguously identify a source. Usually some type of convention is established for creating key values, e.g., the first author's last name in lowercase followed by the year of publication: "chomsky57". Maximum length is 1000 characters. All sources must have a valid key value and this value must be unique among source key values. A valid key value is any combination of ASCII letters, numerals and symbols (except the comma).

A.2.12.21 Key field (`keyField`)

Used for alphabetizing, cross referencing, and creating a label when the author information is missing. This field should not be confused with the source's key attribute. Maximum length is 255 characters.

A.2.12.22 Keywords

Key words used for searching or possibly for annotation. Maximum length is 255 characters.

A.2.12.23 Language

The language the document is in. Maximum length is 255 characters.

A.2.12.24 Location

A location associated with the entry, such as the city in which a conference took place. Maximum length is 255 characters.

A.2.12.25 LCCN

The Library of Congress Call Number. Maximum length is 20 characters.

A.2.12.26 Month

The month in which the work was published or, for an unpublished work, in which it was written. Maximum length is 100 characters.

A.2.12.27 Mathematical Reviews number (mrnumber)

The Mathematical Reviews number. Maximum length is 25 characters.

A.2.12.28 Note

Any additional information that can help the reader. The first word should be capitalized. Maximum length is 1000 characters.

A.2.12.29 Number

The number of a journal, magazine, technical report, or of a work in a series. An issue of a journal or magazine is usually identified by its volume and number; the organization that issues a technical report usually gives it a number; and sometimes books are given numbers in a named series. Maximum length is 100 characters.

A.2.12.30 Organization

The organization that sponsors a conference or that publishes a manual. Maximum length is 255 characters.

A.2.12.31 Pages

One or more page numbers or range of numbers, such as 42–111 or 7,41,73–97 or 43+ (the “+” in this last example indicates pages following that don’t form a simple range). Maximum length is 100 characters.

A.2.12.32 Price

The price of the document. Maximum length is 100 characters.

A.2.12.33 Publisher

The publisher’s name. Maximum length is 255 characters.

A.2.12.34 School

The name of the school where a thesis was written. Maximum length is 255 characters.

A.2.12.35 Series

The name of a series or set of books. When citing an entire book, the title attribute gives its title and an optional series attribute gives the name of a series or multi-volume set in which the book is published. Maximum length is 255 characters.

A.2.12.36 Size

The physical dimensions of a work. Maximum length is 255 characters.

A.2.12.37 Title

The work's title, typed as explained in Kopka and Daly (2004). Maximum length is 255 characters.

A.2.12.38 Type

The value of the OLD source type attribute is the BibTeX entry type, e.g., “article”, “book”, etc. A valid type value is obligatory for all source models. The chosen type value will determine which other attributes must also possess non-empty values, cf. Table A.1.

A.2. OLD objects

type	required attributes
article	author, title, journal, year
book	author or editor, title, publisher, year
booklet	title
conference	author, title, booktitle, year
inbook	author or editor, title, chapter or pages, publisher, year
incollection	author, title, booktitle, publisher, year
inproceedings	author, title, booktitle, year
manual	title
mastersthesis	author, title, school, year
misc	
phdthesis	author, title, school, year
proceedings	title, year
techreport	author, title, institution, year
unpublished	author, title, note

Table A.1: BibTex source types and required attributes.

A.2.12.39 Type field (`typeField`)

The type of a technical report—for example, “Research Note”. Maximum length is 255 characters.

A.2.12.40 URL

The universal resource locator for online documents; this is not standard but supplied by more modern bibliography styles. Maximum length is 1000 characters.

A.2.12.41 Volume

The volume of a journal or multi-volume book. Maximum length is 100 characters.

A.2.12.42 Year

The year of publication or, for an unpublished work, the year it was written. Generally it should consist of four numerals, such as 1984.

A.2.13 Speaker

An OLD speaker object represents a speaker or consultant who is the source of a linguistic form or collection thereof or who is the speaker on a recording.

A.2.13.1 Dialect

The value of the dialect attribute is a string denoting the dialect of the speaker. The value may not exceed 255 characters.

Note that for abstract lexical forms, where it does not make sense to specify a speaker, dialects can be specified via tags—perhaps with a special syntax to facilitate search, e.g., “dialect:dialect_name”.

A.2.13.2 First name

The first name attribute holds the first name of the speaker. A value is obligatory and cannot exceed 255 characters.

A.2.13.3 HTML

The value of the HTML attribute is a string of HTML that is generated by the system using the value of the page content attribute and the markup language specified in the markup language attribute.

A.2.13.4 Last name

The last name attribute holds the last name of the speaker. A value is obligatory and cannot exceed 255 characters.

A.2.13.5 Markup language

The value of the markup language attribute is one of “Markdown” or “reStructuredText”. Markdown and reStructuredText are *lightweight markup languages*. A lightweight markup language is a markup language (i.e., a system for annotating a document) that is designed to be easy to read in its raw form. This value determines which markup-to-HTML function is employed when the system attempts to generate the HTML value from the user-supplied page content value. If no value is specified, “reStructuredText” will be the default.

A.2.13.6 Page content

The value of the page content attribute is a string that can be used to construct a web page for the speaker. Using the markup language specified, the system generates an HTML page for the speaker (which is cached in the HTML value) based on the page content value.

A.2.14 Syntactic category

Syntactic category objects are used to categorize form objects into morphological or syntactic classes.

A.2.14.1 Description

The value of the description attribute can be used to describe the category and/or clarify its intended usage.

A.2.14.2 Name

The name attribute holds the name of the category. Example names might be “N”, “S”, “Agr”, “VP”, “V”, “Noun”, “Sentence”, “CP”, etc. A non-empty value for this attribute is obligatory, must be unique among other syntactic category name values, and may not exceed 255 characters.

A.2.14.3 Type

Syntactic categories are themselves categorized via the type attribute. Valid values are “lexical”, “phrasal” and “sentential”. The purpose of this attribute is to help the system to better understand the categorization. This categorization could be useful for functionality that, say, seeks to induce a grammar of the morphology of the language. The available syntactic category types may change in future versions of the OLD.

A.2.15 Tag

Tags are general-purpose, user-defined objects that can be associated to forms, files and collections. Any form, file or collection may have zero or more tags associated to it. Example usage of a tag would be to create tags for linguistic phenomena relevant to one’s research; searches could then make reference to the presence or absence of this tag.

There are two special tags that are identified by their name values; these are the “restricted” and “foreign word” tags. These tags cannot be deleted via the interface (and should not be forcefully deleted by administrators using the RDBMS as this may have unintended consequences). Objects tagged with the restricted tag can only be accessed by administrators and users who are classified as unrestricted by the active application settings. The foreign word tag is useful for tagging forms that correspond to foreign words. The system uses the foreign word information to create exceptions to transcription validation specifications. For example, if the character *j* is not permitted in orthographic transcription values, having a foreign word form for *John* will cause the system to make an exception for cases where the fieldworker uses English orthography to transcribe *John* within an object language orthographic transcription.

A.2.15.1 Description

The value of the description attribute can be used to describe the tag and/or clarify its intended usage.

A.2.15.2 Name

The name attribute holds the name of the tag. Example names might be “VP ellipsis”, “double object” or “needs verification”. A non-empty value for this attribute is obligatory, must be unique among other tag name values, and may not exceed 255 characters.

A.2.16 Translation

Translation objects represent translations of forms into the metalanguage. A form object can have multiple translation objects. Each translation object has transcription and grammaticality attributes. When a form object is destroyed, so too are its translations.

The grammaticality attribute of a translation is intended to signify both the appropriateness of the metalanguage translation to the object language form as well as the grammaticality of the metalanguage translation. Given this, the OLD should be modified to include sensible defaults for this attribute. For example, * and ? to indicate translations in the metalanguage that are ungrammatical and questionable, respectively, and \neq to indicate a grammatical translation that is inappropriate to a given object language form. A value of *intended* may also be useful here.

A.2.17 User

User objects represent the authorized users of an OLD web service. Authenticating to an OLD web service means supplying values for username and password attributes that match those of an existing user model. Only users with a role value of “administrator” are authorized to create new users. An authenticated user is permitted to update her own user model; however, only administrators can change the value of the username attribute.

A.2.17.1 Affiliation

The value of the affiliation attribute is a string representing the school or institution with which the user is affiliated. A value here is optional. Maximum allowable length is 255 characters.

A.2.17.2 Email

The email attribute holds the email address of the user. A valid email must be provided. Maximum allowable length is 255 characters.

A.2.17.3 First name

The value of the first name attribute is the first name(s) of the user. A value here is obligatory. Maximum allowable length is 255 characters.

A.2.17.4 HTML

The value of the HTML attribute is a string of HTML that is generated by the system using the value of the page content attribute and the markup language specified in the markup language attribute.

A.2.17.5 Input orthography

The input orthography is a reference to an existing orthography object. The purpose of a user-specific input orthography is to allow for the possibility that users will enter form transcriptions (and possibly also morpheme breaks) using one orthography (i.e., their input orthography) but that these transcriptions will be translated into another orthography (i.e., the system-

wide storage orthography) for storage in the database. When outputting the forms, the system would then re-translate them from the storage orthography into the user's output orthography. Previous OLD applications implemented this user-specific orthography conversion server-side. However, with the new architecture of the OLD ≥ 1.0 this added complication seems best implemented client-side.

A.2.17.6 Last name

The value of the last name attribute is the last name of the user. A value here is obligatory. Maximum allowable length is 255 characters.

A.2.17.7 Markup language

The value of the markup language attribute is one of "Markdown" or "reStructuredText". Markdown and reStructuredText are *lightweight markup languages*. A lightweight markup language is a markup language (i.e., a system for annotating a document) that is designed to be easy to read in its raw form. This value determines which markup-to-HTML function is employed when the system attempts to generate the html value from the user-supplied page content value. If no value is specified, "reStructuredText" will be the default.

A.2.17.8 Output orthography

The output orthography is a reference to an existing orthography object. The purpose of a user-specific input orthography is to allow for the possibility that users will enter form transcriptions (and possibly also morpheme

breaks) using one orthography (i.e., their input orthography) but that these transcriptions will be translated into another orthography (i.e., the system-wide storage orthography) for storage in the database. When outputting the forms, the system would then re-translate them from the storage orthography into the user's output orthography. Previous OLD applications implemented this user-specific orthography conversion server-side. However, with the new architecture of the OLD ≥ 1.0 this added complication seems best implemented client-side.

A.2.17.9 Page content

The page content attribute holds a string representing the content of the user's page. This content should be written using the markup language specified in the markup language attribute.

A.2.17.10 Password

When creating a user, a valid value for the password attribute must be supplied. A valid password is composed of at least eight characters but no more than 255. It must contain either at least one printable character not in the printable ASCII range or one symbol, one digit, one uppercase letter and one lowercase letter. For example, "dave.Smith1" is a valid password, as is "philippe.gagné". (The latter contains a non-ASCII character.)

The system stores the password in the database encrypted using the Python PassLib module's implementation of the PBKDF2 key derivation function and the value of the salt attribute. During authentication attempts, the system applies the same encryption to the supplied password values and

authentication succeeds if the encrypted password string from the request matches the encrypted password of the specified user. This means that even administrators of the system are unable to view any user passwords in their unencrypted form.

A.2.17.11 Remembered forms

The value of the remembered forms attribute is a collection of form objects that the user has “remembered”. Note that this attribute is not included in the JSON object representation of user models. Retrieving a user’s remembered forms requires a separate request to the application.

A.2.17.12 Role

The role attribute is used to classify users and is the basis for the authorization functionality. Every user must have a value for the role attribute. Valid values are “administrator”, “contributor” and “viewer”. Administrators have unrestricted access to all requests on all resources, contributors have read and write access to almost all resources and viewers have only read access.

A.2.17.13 Salt

The salt value is a randomly generated UUID that aids in the secure encryption of a user’s password. this value is generated by the system when a user is created.

The randomly generated salt value is appended to the user’s password and then the whole thing (i.e., password plus salt) is hashed—i.e., trans-

formed into a cryptographic hash value, i.e., a fixed-size string from which the original password cannot be deduced—before being saved to the database. During an authentication attempt, the system again appends the salt to the password entered, hashes the whole thing, and checks whether the two hashes match. If they do, the authentication attempt has succeeded. Hashing passwords is an important security precaution because it means that user passwords are not stored in plain text in a database.

Using a randomly generated salt for each user makes it more difficult for a malicious person who has gained access to a database of hashed passwords to crack those passwords. A standard method of cracking passwords is to use a table of commonly used passwords that are pre-hashed. Once the attacker finds a match using this table, they know what the unencrypted value of the password is. However, if a unique salt is used for each user password, then the attacker must generate a new hash table for each salt/user and this is a more difficult task. A good overview of the reason for using salts and hashing passwords is available at <http://culttt.com/2013/01/21/why-do-you-need-to-salt-and-hash-passwords/>

A.2.17.14 Username

The value of the username attribute is a string consisting of letters of the English alphabet, numbers and the underscore. Each user must have a unique username value and no two usernames may be the same. Only an administrator can update the username of a user model. Maximum allowable length is 255 characters.