Efficient Extraction of Ontologies from Domain Specific Text Corpora

by

Tianyu Li

B.Eng., Zhejiang University, 2009

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

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

December 2011

Abstract

There is a huge body of domain-specific knowledge embedded in free-text repositories such as engineering documents, instruction manuals, medical references and legal files.

Extracting ontological relationships (e.g., ISA and HASA) from this kind of corpus can improve users' queries and improve navigation through the corpus, as well as benefiting applications built for these domains.

Current methods to extract ontological relationships from text data usually fail to capture many meaningful relationships because they concentrate on single-word-terms or very short phrases. This is particularly problematic in a smaller corpus, where it is harder to find statistically meaningful relationships.

We propose a novel pattern-based algorithm that finds ontological relationships between complex concepts by exploiting parsing information to extract concepts consisting of multi-word and nested phrases.

Our procedure is iterative: we tailor the constrained sequential pattern mining framework to discover new patterns. We compare our algorithm with previous representative ontology extraction algorithms on four real data sets and achieve consistently and significantly better results.

Table of Contents

List of Tables

List of Figures

Glossary

POS Part of Speech

LASER LArge ScalE Relation extraction system

ESPRESSO The ESPRESSO algorithm

GHC The Guided Hierarchical Clustering algorithm

SCIP Subsumption Candidate Instance Pair

LCA the Lowest Common Ancestor

SPM Sequential Pattern Mining

PMI the Point-wise Mutual Information

Acknowledgments

I want to thank my supervisors Rachel Pottinger and Laks V.S. Lakshmanan for all their help during my master study period, especially those discussions we had for my thesis project. I am always inspired by their guide and ideas.

I also want to thank the NSERC BIN funding for supporting me on this ontology extraction research project, it gives a great opportunity to learn more about my research area and sharpen my technical skills.

Finally, I would like thank my mom, who always supports me from China, which is 9,000 km away from here, and gets me through all good times and bad times.

Chapter 1

Introduction

An ontology is a specification of conceptualizations in a specific domain [3]. An ontology typically includes, at a minimum, concepts and hierarchical relationships among the concepts. The two fundamental hierarchical relationships are ISA, asserting a class-subclass relationship or class-instance relationship, which forms an ontology's taxonomic backbone, and HASA, asserting a whole-part relationship.

Building an ontology from unstructured text on the web can bridge the gap between human-readable data and machine-readable knowledge in a specific area by improving navigation and information discovery. However, it is not enough to simply reuse general-purpose ontologies (e.g., WordNet). Such ontologies have limited coverage, particularly in specialized fields, where jargon and terminology can have very different meanings from their senses in a more general domain. For example, "foreign materials" in Architecture usually refers to substances on the surface of piping equipments, quite different from the meaning of foreign in "foreign nationals" which may be found in a legal document; and "agent" means different things in the medical and law domains.

Previous works on ontology discovery from unstructured text either explore hierarchical relationships among concepts from their distribution in a corpus [5, 27], or use lexico-syntactic patterns [21] (e.g., Hearst patterns [14]). Unfortunately, the former approaches usually suffer from data *sparsity*, which is common in small corpora. While the latter approaches may not suffer from this, both approaches suffer from a second drawback: they generally extract *single-word terms* or *very*

short phrases as concepts. This becomes a problem when dealing with sentences involving complex nested noun phrases. Thus these *previous approaches fail to capture many meaningful and complex concepts in a domain, with the result that the discovered ontology is usually trivial or redundant given a general thesaurus*.

We illustrate these ideas with an example drawn from a real architecture web dataset, which is one of the web datasets we use in our experiments:

> avoid contact with finishes which may radiate noise, such as the concrete structure, future framing and drywall

is a text fragment from this data set. In this fragment, "finishes which may radiate noise" is a meaningful concept, and intuitively there is an ISA relationship from each of "concrete structure", "future framing" and "drywall" to it respectively. However, both approaches above fail to capture this because:

1. This phrase is not frequent enough to be considered a concept by statistics-based algorithms.

2. Pattern-based algorithms identify noun phrases by matching Part of Speech (POS) tags applied by a tagger; hence a phrase with clause modifier cannot be extracted and identified as a concept.

3. Finally, "finishes which may radiate noise" is a nested noun phrase, so a naive pattern-based algorithm only looking for patterns like "such as" may end up retrieving "noise" as the parent concept in this ISA relationship, which is obviously wrong!

Extracting rich and complex concepts along with ontological relationships among them as accurately as possible is the main goal of this work. To this end, we develop an iterative pattern-based algorithm called LArge ScalE Relation extraction system (LASER). The central paradigm used by LASER is an iterative framework of starting with seed patterns that signal an occurrence of (ISA or HASA) relations, which are used to extract instances of relations in the corpus. They are in turn used to induce more patterns from the corpus and so forth. At every stage, the extracted patterns and instances are scored, reflecting their degree of reliability. When the score of extracted patterns drops below a threshold, the process terminates. This

iterative framework was first pioneered by Brin [4] for extracting general relations from the web and was adapted by researchers to various other contexts. Indeed, The ESPRESSO algorithm (ESPRESSO) proposed in Pantel and Pennacchiotti [21] adapted this framework for ontology extraction. While a more detailed comparison with the ESPRESSO approach appears in Chapter 2, some fundamental differences between LASER and ESPRESSO include the following.

1. ESPRESSO assumes a set of seed instances for its iteration while LASER starts instead with a set of seed patterns well known to be reliable in the ontology extraction literature [2, 12, 14].

2. We extract noun phrases by analyzing the parse tree as opposed to relying on POS tag matching, a method most previous algorithms including ESPRESSO rely on.

3. Nested noun phrases pose a serious challenge for the correct extraction of relations. Unlike previous algorithms, we solve this problem effectively.

4. We solve pattern discovery using a novel approach based on constrained closed sequential pattern mining.

We make the following contributions.

- We start with a list of reliable seed patterns and find ISA/HASA relationships and new patterns that signal occurrences of such relationships.

- We extract the noun phrases having ISA/HASA relationships in between from the constituent parse tree of the text matching the patterns, and use a novel algorithm to resolve the challenge of determining the correct noun phrases linked by these relations when nested noun phrases are present.

- We tailor the closed sequential pattern mining framework to find frequent patterns that signal ISA/HASA relationships, and generalize the pattern to allow the introduction of more new distinct patterns.

- We carry out a detailed experimental comparison between LASER and major representative algorithms from previous work on four real web datasets. Our

experiments show that our algorithm has a significantly better recall than previous algorithms while enjoying a comparable or better precision. In particular, our F_β-measure is significantly better than previous algorithms, for $\beta = 0.5, 1, 2$, demonstrating our approach's superiority. We show that our algorithm works very well and is stable on both small and large corpora. We also show that our algorithm is much more scalable than previous ones.

The rest of paper is structured as follows: Chapter 2 classifies and describes related work. Chapter 3 introduces our system architecture and components in detail. We discuss the LASER approach and the algorithms in Chapter 4 and discuss our experiments and lessons learned in Chapter 5. Finally, Chapter 6 concludes and discusses future work.

Chapter 2

Related Work

Ontologies fall into one of three types [3]:

1. A *formal ontology* is a set of logical expressions relating concepts by axioms and definitions; as such it can support inference and computation.

2. A *prototype-based ontology* typically lacks labels for concepts, but relies on term clusters, which are regarded as prototypical instances of the underlying concept or category.

3. A *terminological ontology* describes concepts by concept labels or synonyms, instead of using prototypical instances, and expresses hierarchical (e.g., subtype-supertype and whole-part) relationships between concepts.

This paper focuses on automatically building terminological ontologies from domain-specific text corpora. For example, given a set of engineering documents containing "furnace", "air conditioning unit", and "HVAC", we would try to find that the air conditioning unit ISA HVAC and the furnace ISA HVAC. In contrast, a prototype-based ontology might cluster "furnace" and "air conditioning unit" together, but it would not say that either ISA "HVAC", and a formal ontology would include additional semantic relationships. We survey previous related work below.

There are three main approaches to learning ontologies from unstructured text [3]:

Data mining: Clustering approaches cluster similar words based on the hypothesis that similar words tend to occur together in similar context [13]. Some

approaches assign labels to the clusters, treating the labels as concepts and the terms in the cluster as its instances. More recently, association rule mining has been used for ontology learning [20, 27].

Lexico-syntactic patterns: Lexico-syntactic patterns such as Hearst Patterns [14] are used to extract relationships between terms. For example, the pattern "X such as Y" frequently implies Y ISA X.

***Web as a data source*:** To overcome data sparsity, some algorithms use the web as an additional data source, possibly in conjunction with other data sources like WordNet[1].

Most approaches build ontologies consisting of single words [5, 22] or short common compounds [6, 27]. Indeed, very few works allow for longer and complex terms to be concepts, generally because they have a very low frequency of occurrence compared to short ones. Drymonas et al. [10] makes an attempt in this direction and allows more complicated noun phrases. They use a statistical measure to select meaningful concepts. Then an agglomerative clustering algorithm is applied on these concepts to build a prototype-based ontology. This system selects concepts before building ontology, however, its way of deter- mining important noun phrases as concepts can be applied to our method to improve the extracted instance quality, which is a promising future work for us.

Data mining: Most clustering-based algorithms first produce prototype-based ontologies. Some then additionally assign labels to the clusters. Caraballo [5] builds an unlabeled hierarchy of nouns using bottom-up clustering on the cosine similarity between nouns' occurrences in documents. For leaf clusters, he assigns a label using syntactic patterns, while for internal clusters he assigns the most dominant hypernym of the largest number of the node's descendants.

Pantel and Ravichandran [22] first cluster words from a text corpus; each cluster forms a concept. They extract concept names by searching for syntactic patterns such as "concept apposition-of instance" and "concept such as instance". The word that co-occurs with the most dominant instances in a cluster most frequently is picked as the concept name. They create ISA relationships between the concept

[1] http://wordnet.princeton.edu/

and all instances in the cluster. This helps with data sparsity since not all instances need to co-occur with the concept name for us to derive the ISA relationship between them. However, this kind of ISA relationship is necessarily confined to one level hierarchy, while our work produces a complex multi-level concept hierarchy.

Sanderson and Croft [27] use association rules to find ISA relationships (called subsumption in the paper) between terms. They use the intuition that for two terms, x and y, x ISA y holds if $P(x|y)$ is sufficiently large (e.g., 0.8) and $P(y|x) < 1$, where $P(x|y)$ is the probability that a document contains x given that it contains y. This approach is purely based on statistical heuristics, which cannot produce a high-quality ontology alone. We use statistical heuristics as well as syntactic and semantic (parsing) information.

Lexico-syntactic patterns: Pure pattern-based methods (e.g., Pantel and Pennacchiotti [21]) usually iteratively interleave pattern discovery and instantiation until the reliability drops below a threshold. These methods tend to suffer from low recall, especially in a smaller scale corpus.

ESPRESSO [21] is the system most related to ours. Given a small set of seed instances for a particular relation, the system learns lexical patterns, applies them to extract new instances, and then uses the web to filter and expand the instances. This procedure continues iteratively until it meets some stopping criteria such as reliability dropping below a threshold. We adapt their scoring mechanism in the iterative process, but instead of starting with seed instances, we bootstrap from seed patterns, which are more reliable.

We extract instances from patterns substantially differently from ESPRESSO as well: we rely on deep parsing information to get richer concepts that cannot be identified by regular expression matching over data obtained from shallow parsing. Our pattern finding is also generalized to be less restrictive and more expressive than ESPRESSO. Additionally, we provide a novel formulation of pattern discovery as a constrained sequential pattern mining problem.

Hybrid approaches: As the name suggests, these approaches borrow ideas from one or more of the previous approaches. The Guided Hierarchical Clustering algorithm (GHC) [6] is a representative one in this class. It first calculates the similarity between a set of given input terms based on syntactic dependency features in the corpus including adjective modifiers, prepositional phrase modifiers and noun

phrases in subject or object position. Then, using an agglomerative clustering algorithm, it picks the most similar pair of terms in the remaining list of pairs to be clustered, and uses WordNet, Hearst patterns in the corpus, and the WWW to position them in the growing ontology. If no relationship is found, the pair is clustered. Finally, they make sure the resulting ontology is a connected hierarchy. Unlike these, our approach finds relationships before concepts. We start from patterns that indicate relations, and then get concepts from there, thus not requiring terms as input.

Zavitsanos et al. [35] use topic modeling to extract concepts which are represented as distributions of words. TextToOnto [18] and Text2Onto [7] focus on conceptual relationships rather than hierarchical relationships. The concept hierarchy is used as a knowledge base to find more complex relations. For practical applications, algorithms that produce concrete ISA and HASA relations with labeled concepts are more useful than ones that produce latent topics and word clusters without labels.

Formal Ontologies: YAGO [30] automatically creates a formal ontology by extending WordNet by leveraging Wikipedia's info boxes. It reports very high reliability. More recently, SOFIE [31] extends YAGO by extracting relationships from free text. Thus, they can process Wikipedia articles (not just info boxes) and indeed arbitrary web pages. Kylin/KOG [33, 34], DBPedia [1], and DBLife [9] are other examples of systems that have extracted very large ontologies containing millions of entities and relations. A series of recent papers have followed a declarative approach to information extraction [25, 28].

One of the distinguishing features of our work is that our algorithm can produce high quality ISA and HASA relations from domain specific text corpora. Our experiments show that our algorithm significantly outperforms previous ones both in quality and in running time. Indeed, as mentioned in Suchanek et al. [31], even for systems generating formal ontologies, algorithms that produce hierarchical relations on a large scale and with a high quality are essential in order to give the resulting ontology a clean structure.

Poon and Domingos [24] induce and populate a probabilistic ontology, using dependency-parsed text as input. The output ontology of this system mainly consists of verb classes in hierarchy with nouns as their argument class, and ISA re-

lationships between verb classes and HASA relationships between the verb classes and their arguments.

Chapter 3

System Architecture and Background

This chapter outlines the architecture of our system and develops the key notions and ideas used in our algorithms.

3.1 LASER System Architecture

LASER (Figure 3.1) uses an iterative process. The dotted circle in Figure 3.1 highlights the main components.

LASER takes as input the preprocessed corpus consisting of a set of text documents, with each word tokenized. Additionally, it takes in a set of *seed patterns*, i.e., lexico-syntactic templates such as Hearst patterns [14] that imply ISA/HASA relationships. We define a Subsumption Candidate Instance Pair (SCIP), as a pair of noun phrases x, y such that they are involved in a class-subclass or class-instance (ISA) or a whole-part (HASA) relationship, and denote it SCIP (x, y). It states that either HASA(x, y) or ISA(x, y) holds.

The LASER system has the following modules:

0. Corpus Text Parsing and Indexing: We generate the parse tree of the corpus text with a parser such as the Stanford Parser[1], and build an inverted index on the corpus text as well as on the parse tree for efficient lookup. This module is run

[1] http://nlp.stanford.edu/software/lex-parser.shtml

once for a given corpus.

1. ISA/HASA **Pattern Instantiation**: This module finds the sentences containing seed pattern instances and extracts noun phrases involved in the ISA/HASA relationships.

2. **SCIP Extension & SCIP Scoring**: This module extends and ranks the instance pairs so that the highest ranking pairs are considered as seed instance pairs to aid new pattern discovery.

3. Frequent Pattern Discovery: This module takes the seed instance pairs from module 2 and uses them to find new patterns that imply ISA or HASA relationships.

4. Pattern Scoring: The candidate patterns from the previous module are scored to select seed patterns for the extraction of new instances in the next round.

This iterative process of finding instances and patterns continues until LASER cannot find new instances or the new patterns' score discovered drops below a threshold, which is tuned empirically. In our experiments we used the threshold of having the average score of patterns discovered in the current iteration being above 50% of the average score of patterns found in the previous iteration.

3.2 Background

We make use of the following notions in the LASER system.

Definition 1 *A* head word *is the word that determines the syntactic type of the noun phrase of which it is a member, while other words modify the head.*[2] □

Definition 2 [19] *A* noun phrase *is a syntactic unit of the sentence where information about the embedded noun is gathered. Therefore the noun is the* head word *of the noun phrase, the central constituent which determines the syntactic characteristics of the phrase. A noun phrase usually consists of an optional determiner, zero or more adjective phrases, a noun head and some other clause modifiers.* □

In the example in the introduction, "finishes which may radiate noise" is a noun phrase with head word "finishes". A noun phrase is clearly a concept in the corpus and is a candidate concept in the domain-specifc ontology that we seek to build.

[2] See http://en.wikipedia.org/wiki/Head_(linguistics)

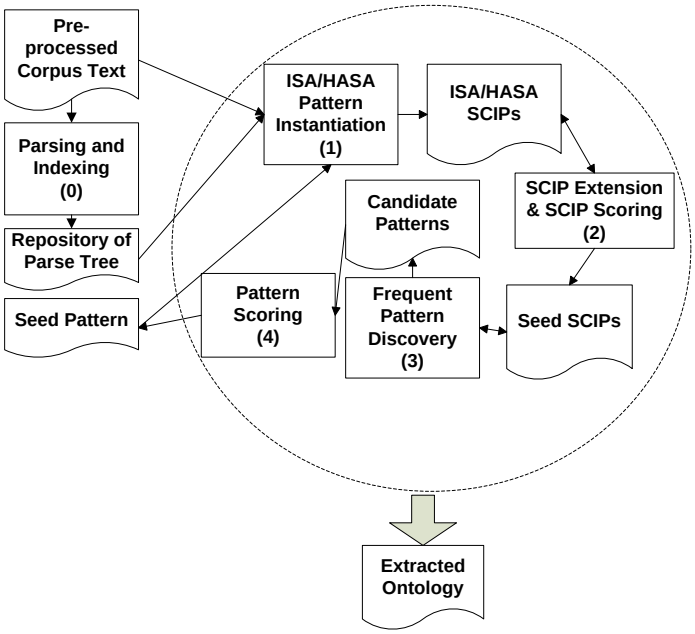

Figure 3.1: Ontology Extraction System Architecture

Definition 3 *A* pattern *is a sentence fragment of the form: $NPList_1$ Connector $NPList_2$ where each $NPList_i$ is a list of one or more noun phrases and Connector is a sequence of words in a short phrase and/or the corresponding POS tags, which signals a relationship between the concepts represented by $NPList_1$ and $NPList_2$.* □

For ISA and HASA relations, either $NPList_1$ or $NPList_2$ is the parent (i.e., the more general concept in an ISA relationship and the "whole" concept in a HASA relationship) while the other NPList is a child or a list of children (the more specific concept in an ISA relationship and the "part" concept in a HASA relationship), as determined by the specific connector. After identifying $NPList_1$ and $NPList_2$ of a pattern in the text corpus, we generate candidate instance pairs for this pattern consisting of one noun phrase from $NPList_1$ and another from $NPList_2$.

As an example, in the clause "plumbing equipment such as steel storage tanks, pressure reducing stations and ductile iron pipe", the string "such as" acts as a connector; "plumbing equipment", "steel storage tanks", "pressure reducing stations"

and "ductile iron pipe" are noun phrases. In this example, $NPList_1$ is a single noun phrase and $NPList_2$ is a list of three noun phrases. The connector "such as" indicates each noun phrase in $NPList_2$ is an instance or subclass of the noun phrase $NPList_1$, that is, one can infer the relations ISA(plumbing equipment, steel storage tanks), ISA(plumbing equipment, pressure reducing stations) and ISA(plumbing equipment, ductile iron pipe).[3] Recall, as defined in Section 3.1, we use the term SCIP pair, denoted $SCIPs(x,y)$, to indicate a pair of noun phrases x,y that are related by an ISA or a HASA relationship.

This kind of pattern was first proposed by Hearst [14] and extended by many subsequent papers. We use the seven patterns below, called *seed patterns*, as the initial ISA patterns in the first round of our iterative taxonomic relation extraction procedure:

1. NP_0 such as $NP_1, NP_2, ..., NP_{n-1}$ (and|or) NP_n
2. such NP_0 as $NP_1, NP_2, ..., NP_N$ (and|or) NP_n
3. $NP_1, NP_2, ..., NP_n$ (and|or) other NP_0
4. NP_0, (including|especially) $NP_1, NP_2, ..., NP_{n-1}$ (and|or) NP_n

The above four are from Hearst [14] while the following three are from Cimiano and Staab [6], which extends Hearst [14]:

5. NP_1 is NP_0
6. NP_1, another NP_0
7. NP_0 like NP_1

As a follow up to Hearst's work [14], Berland and Charniak [2], Girju et al. [12] proposed similar lexico-syntactic patterns implying part-whole relationships (i.e., HASA relationships). We used five patterns from their work as our initial list of HASA patterns:

1. NP_0 (consists | consist | made) of NP_1

[3] ISA(x,y) indicates y is an instance (or subclass of) x.

2. NP_1 (members | a member | a part) of NP_0

3. NP_0 have | has NP_1

4. NP_1 inside NP_0

5. parts of NP_0 include NP_1

Chapter 4

Ontology Extraction

In this section, we provide a detailed description of modules 1–4 in the architecture schematic of Figure 3.1 and provide the key algorithms used in our LASER system. Specifically, we describe the algorithms for correct handling of nested noun phrases and for pattern discovery using constrained closed sequential itemset mining.

4.1 ISA/HASA Pattern Instantiation

This module takes a list of known patterns suggesting ISA or HASA relationships and applies the patterns to the input corpus to find sentences matching the patterns.

4.1.1 Extracting Noun Phrases

Previous works [6, 21] usually find pattern instances by matching each POS tagged sentence with regular expressions.

For example, the regular expression $(DT \setminus t(\setminus w+))?(JJ \setminus t(\setminus w+))?((NN(S?) \setminus t([a-z]+) \setminus s?)+)$ determines a non-recursive noun phrase, in which zero or one determiners (DT) followed by zero or one adjectives (JJ) plus one or more singular or plural nouns ($NN(S?)$) is a noun phrase. Thus, "a stringent requirement", which is tagged as "DT a JJ stringent NN requirement", can be recognized as a noun phrase because it matches the regular expression.

Such a strategy has the following limitations:

1. The simple POS tag rules may identify the wrong noun phrase because the

context is not considered. For example, in the sentence "Adding flooring finishes such as carpet can significantly change the Apparent-IIC", "flooring finishes" is the correct parent noun phrase of "carpet". However, according to the POS tags "Adding/VBG flooring/NN finishes/NNS" and the POS based rules that recognize the "VBG" as an adjective modifier, "adding flooring finishes" (a verb phrase) is incorrectly identified as a noun phrase, leading to the incorrect inference ISA(carpet, adding flooring finishes).

2. Strict application of pattern matching may fail to capture some patterns that contain the proposed patterns. For example, suppose we try to extract SCIPS with the seed pattern "... including ...". Then even though the connector "... including but not limited to ..." contains the seed pattern and is meaningful, we cannot extract SCIPS from it because the pattern matching requires that a noun phrase immediately follow the connector "including".

3. Simple POS tag rules cannot identify some noun phrases that have complex structures using modifiers. For example, "coatings that may be detrimental" is a noun phrase occurring in one of our real data sets. It has an attributive clause as a modifier, which cannot be correctly identified by simple regular expression rules.

To overcome these limitations, we perform pattern matching by first matching sentences containing lexical connectors, and then extracting the corresponding noun phrases from the text segments either surrounding those connectors or in between them, by analyzing the constituent parse tree structure for the sentences. The idea is that a well-trained parser like the Stanford Parser can be more effective at determining noun phrases than simply matching regular expressions over POS tags. For example, for the pattern $NPList_1$ *such as* $NPList_2$, we extract noun phrases from the left of and the right of "such as" respectively. We achieve this by identifying appropriate noun phrases in the parse tree of the matched sentence, knowing the position of the connectors in that tree. This novel approach is in contrast to previous approaches which use hand-crafted rules to match a whole sentence.

However, LASER adopts a slightly different strategy for the first iteration of pattern instantiation and the later iterations: it extracts noun phrases with the matched

words in the pattern as clear boundary in first run but allows a sliding window around the words for patterns in later run. For example, this pattern

$$NP_0 \text{ such as } NP_1, NP_2, ..., NP_{n-1} \text{ (and|or) } NP_n$$

is used in the first iteration. LASER will look for the noun phrase exactly preceding "such as" in the parse tree as the parent noun phrase. Similarly, the child noun phrases are extracted with the matched position of "such as" and "and|or" as clear boundaries.

In later rounds, LASER allows the noun phrases to occur around the pattern with a small gap, which is $w - wordsizeofpattern$. LASER uses a sliding window to simulate the procedure of looking from closer matches to further away ones. For example, "$,/,including/VBG$" is a pattern LASER found in later iterations, so LASER uses a sliding window of word size w that its left end starts with "$,/,$" and moves towards left (one word at a time), until the right end of the window reaches "$including/VBG$". In this way LASER tries to find a noun phrase with the last word that is on the left of the pattern and resides in the window. Similarly LASER moves the window towards right until the left end of the window reaches "$,/,$", starting with the right end of the window placed at "$including/VBG$", and finds the corresponding child noun phrases.

The reason we adopt different strategy in later rounds is: the seed patterns used in first iteration are defined by linguistic experts. So the words in these kinds of patterns are more meaningful in terms of semantic, so that we use exact match to get more accurate result. However the generic patterns in later iterations are found by frequent substrings, which can be a part of a meaning connector that is not frequent enough. Take the example of ", including" and ", including but not limited to", the relaxed match gives some longer patterns a second chance.

4.1.2 Nested Noun Phrase Challenge

One challenge in ontology extraction is that noun phrases may be nested in another noun phrase. In this case it is difficult to identify the appropriate noun phrases in the extracted relationship. Below, we give two examples that illustrate this challenge: in Example 1 the shorter, nested noun phrase is the correct one. In Example 2, the

longer, outer noun phrase is the correct one. The examples are taken from one of the real data sets described in Section 5.1.

Example 1 *Consider the sentence "Provisions of shading devices,* such as *overhangs or vertical fins, to let in quality natural light but exclude undesired direct sun light should be considered ." Here, "Shading devices" inside "provision of shading devices" is a nested noun phrase.*

"Shading devices" is the correct parent concept (the more general concept) of the ISA *relationship not "provision of shading devices." That is, in making inferences about* ISA *relationship we should be using "shading devices", not "provision of shading devices." In this case, the shorter noun phrase is the correct choice.* □

Example 2 *In "The work shall be carried out in accordance with the authorities having jurisdiction,* including *Ministry of Environment and the Workers Compensation Board of British Columbia and by contractors experienced in this specialty ," the noun phrase "the authorities having jurisdiction" contains the nested noun phrase "jurisdiction".*

In this case the longer phrase is the right choice for use in the relationship inference. Using more complex noun phrase adds an extra dimension to the problem: not only do we aim to avoid outright incorrect choices, but we strive to pick the best among the correct ones. □

These two examples are in sharp contrast and clearly illustrate the challenge in determining the appropriate noun phrase for relationship inference; it is not always better to use the longer noun phrase nor always the shorter one. To solve this challenge, we employ a linguistically based heuristic approach that uses hints from an external source, e.g., a general thesaurus like WordNet. A useful cue about the type of a noun phrase can be obtained from its head word.

For example, the head word for "shading devices" is "devices" and "provision of shading devices" has the head word "provision". When the sentence contains an ISA or HASA pattern but the potential parent noun phrase is nested, such as in Example 1 and in Example 2, we can identify the head words of child noun phrases and the potential parent noun phrases (generated by extracting all noun phrases

from the nested noun phrase recursively), and try to find relationships among these head words in WordNet.

In order to measure the relatedness between words, we use the semantic similarity defined [23], which makes use of corpus statistics and the hierarchical structure in WordNet.

The WordNet::Similarity module[1] implements different variations of semantic similarity.

In our work, we use three of them and take the average:

1. "Path" is the inverse of the shortest path length between two concepts in WordNet.

The other two measures are based on *information content*, a corpus-based measure in information theory that is proposed by Resnik [26] to represent the specificity of a concept (more specific the concept is, larger this value will be). One way to estimate this value is by corpus statistics and the WordNet::Similarity module has pre-computed it for concepts in WordNet using standard corpus.

3. "JCN" is the semantic similarity described by Jiang and Conrath [16], which subtracts the information content of the Lowest Common Ancestor (LCA) of the two concepts, from the sum of the information content of these two, then takes the inverse of the substraction result (convert the distance to similarity).

4. The final measure is "LIN", which is proposed by Lin [17], divides the information content of the LCA by this sum mentioned earlier.

Algorithm 1 extracts the best possible choice for a parent concept given a nested noun phrase (for parent) and a list of noun phrases (for child).

Lines 4 to 12 in Algorithm 1 calculate the sum of the similarity between each candidate parent's head word and head words of all children. We remember the candidate parent that has the maximum similarity sum, *MaxSimSum*.

For example, "Provisions of shading devices, such as overhangs or vertical fins", has two candidate head words: "provision" and "devices" for the candidate parents ("provision of shading devices" and "shading devices", respectively).

[1] A module that implements a variety of semantic similarity and relatedness measures based on information found in the lexical database WordNet.

Algorithm 1 Parent_NP_Resolution_in_Nested_NP

Require: A nested noun phrase (*NestedNP*) containing the potential parent noun phrase; A set of child noun phrases (*ChildList*).

Ensure: The appropriate parent noun phrase (*ParentNP*), which is a hypernym of the noun phrases in *ChildList*.

1: *ParentList* $\leftarrow$ Recursively extract a list of noun phrases containing the last word in *NestedNP* from the parse tree
2: $MaxSimSum = -1$
3: $CurrentCandidate = null$
4: **for all** $Candidate \in ParentList$ **do**
5: $SimSum = \sum_{ChildNP \in ChildList} Similarity(headof(Candidate), headof(ChildNP))$
6: **if** $SimSum > MaxSimSum$ **then**
7: $CurrentCandidate = Candidate$
8: $MaxSimSum = SimSum$
9: **else if** $SimSum == MaxSimSum$ and $length(Candidate) < length(CurrentCandidate)$ **then**
10: $CurrentCandidate = Candidate$
11: **end if**
12: **end for**
13: **if** $MaxSimSum == 0$ **then**
14: Return *ParentNP* $\leftarrow$ shortest *Candidate* in *ParentList*, breaking ties in favor of a candidate with a plural head word if any, and then arbitrarily.
15: **end if**
16: Return *ParentNP* $\leftarrow CurrentCandidate$

Hence, we sum the semantic similarity between "provision" and "overhangs" and between "provision" and "fins". Similarly we sum the semantic similarity between "devices" and "overhangs", "devices" and "fins". In this case, the similarity sum is larger for "devices" than for "provision", which suggests that "shading devices" is a better choice of parent concept.

If two candidates have the same sum, we will choose the shortest one (Line 9–11), because the head word of a parent phrase tends to be closer to the child phrases that specify this parent. When the maximum similarity sum is zero, meaning head words are not found in WordNet (which is possible when we are dealing with a domain-specific corpus), we will try to find the shortest noun phrase, with head word in plural form if it exists, as a default behavior (Line 13–15).

We evaluate this heuristic on a small sample of data, which is the first iteration of ISA SCIP extraction on one of our datasets. In all 51 cases where the nested parent noun phrase challenge exists, this semantic heuristic achieve an accuracy of choosing 90.2% correct parent, while the simple heuristic that always choosing the closest candidate has accuracy of 66.7%, which proves the effectiveness of using semantic similarity to choose correct parent noun phrase.

4.2 SCIP Extension

We can extend the set of SCIP pairs derived by generating several more SCIPS that exploit the inherent ISA relationship between a complex phrase and its head word and the transitivity of ISA relationship. E.g., consider the SCIP ISA(plumbing equipment, ductile iron pipe). We can extend this by generating the SCIP ISA(equipment, ductile iron pipe). Many existing algorithms make the assumption, that if ISA(NP_1, NP_2), then necessarily ISA($head(NP_1)$, $head(NP_2)$). This is an assumption, not necessarily a valid inference. Notice that ISA($head(N_1), head(NP_2)$) does *not* follow from transitivity. In the above example, it turns out ISA(equipment, pipe) *happens* to be valid.

Our observations on real data sets indicate that this assumption results in many erroneous relationships or trivial relationships that can be found in a general ontology. This is because in many cases, the sense of the head word cannot be disambiguated without modifiers. According to our preliminary results, 49% of head word pairs derived do not form valid ISA pairs. For example, following this assumption on ISA(points of penetration of the vapor barrier jacket, raw edges) yields ISA(points, edges), which is meaningless!

In summary, we extend every extracted pair ISA(NP_1, NP_2) from a SCIP by generating the additional pair ISA($head\ (NP_1), NP_2$). This is shown in Algorithm 2. Then we calculate reliability scores for all extracted and extended pairs based on the scoring mechanism described in Section 4.4. Finally, we filter those pairs with scores smaller than average and pick the top ones as seed SCIPS for discovering new patterns in the next iteration.

Algorithm 2 ISA_SCIP_Extension

Require: ($InputSCIP$): a pair of NPs $n1, n2$ s.t. ISA($n1, n2$)
Ensure: (E) is a set of NP pairs having ISA relationship within each pair
1: add $InputSCIP$ to E
2: **if** $headof(ParentNP) \neq ParentNP$ **then**
3: add $(headof(ParentNP), ChildNP)$ to E
4: **end if**

4.3 Frequent Pattern Discovery

Using the seed SCIPS (ISA/HASA relationships) produced by module (2) from Figure 3.1, we want to find new patterns that imply these relationships.

We adopt a *Frequent-Substring-based Pattern Extraction* approach to achieve this new pattern discovery. The idea is to find substrings that frequently occur in between the parent concept and the child concept of a SCIP in the corpus. Using seed instances in the form of SCIP (NP_1, NP_2) as input, and we find co-occurrences of NP_1 and NP_2 in the corpus where the text in between NP_1 and NP_2 is shorter than a pre-defined limit.

To represent this in Datalog, for each sentence I containing both NP_1 and NP_2, we have

$$candConn(X) :- SCIP(NP_1, NP_2) \& contains(I, NP_1, X, NP_2) \\ \& length(X, L) \& L \leq windSize \quad (4.1)$$

$candConn(X)$ means X is a text fragment in which we want to find patterns, if X is the text sequence linking NP_1 and NP_2 in the sentence I, and its length L is smaller than a predefined constant $windSize$.

After collecting text sequences for each SCIP, we find frequent substrings from them.

ESPRESSO [21] finds frequent substrings that contain both concepts of a SCIP, by building a suffix tree for all tagged sentences containing both concepts of instances. This suffix tree keeps a record of all substrings of these sentences. The frequent substrings are considered to be candidates for new patterns.

For example, given a tagged sentence: "Sensory/JJ aspect/NN such/JJ as/IN

air/NN quality/NN can/MD easily/RB be/VB compromised/VBN ./.", and it is given ISA(sensory aspect, air quality). ESPRESSO replaces the actual parent and child concept by "X" and "Y", which gives the generalized tagged sentence : "X such/JJ as/IN Y can/MD easily/RB be/VB compromised/VBN ./." A frequent substring of tagged sentences like above will look like "X such/JJ as/IN Y", in which ISA(X, Y).

However, this kind of pattern is not general enough because ESPRESSO requires *exact* matches of both words and the corresponding POS tags. Due to data sparsity, a problem that is especially severe for a small-scale and domain-specific corpus, frequent patterns are hard to find and the resulting patterns will have limited power in picking out instances in later iterations of instance extraction. ESPRESSO tries to generalize this kind of pattern by replacing terms (their counterpart of our "noun phrase"), that are identified by regular expression matching over tagged sentences, with a uniform symbol "TR". Another example pattern with generalization is "X such/JJ as/IN Y ,/, TR" where TR is a terminology.

As we show in Chapter 5, this kind of pattern still suffers from low recall.

In contrast, instead of just finding frequent substrings in all sentences (which can be really long) containing a SCIP, we find frequent substrings from the text in between the two concepts of a SCIP, which corresponds to the $candConn(x)$ in Equation 4.1. Similarly, we do not require exact matches of both words and their corresponding POS tags. Here is the representation we will use for our problem definition:

Definition 4 A string that satisfies Formula Equation 4.1 is called *candidate connector*, its lexical layer is
$< word_1, word_2, ..., word_n >$
and its syntactic layer (POS Tagging) is
$< POS_1, POS_2, ..., POS_n >$
The combination representation is
$< (word_1, POS_1), (word_2, POS_2), \ldots, (word_n, POS_n) > .$
We want to find frequent substrings of combination representations for *candidate connector*s, as candidate patterns. □

However instead of requiring strict substring of the combination representation,

we allow the existence of wildcard in either lexical layer or syntactic layer so that the pattern is general enough.

Consider the input candidate connector "and others;" "or others" may also be a valid connector but does not occur frequently enough. Instead, we may look for a pattern like "* others", where we require that both terms have the same part of speech tags as the original pattern. Similarly, we could also allow for generalizing patterns based on POS tags.

We solve the above problem using classical closed Sequential Pattern Mining (SPM) [29]. In SPM, given a database of lists (sequences) of transactions (itemsets) ordered by transaction time, the problem is to determine frequent sequential patterns that have a minimum user-specified support, i.e., number of sequences containing the pattern.

In our case, the combination representation of a *candidate connector* can be transformed into a sequence according to the following definition.

Definition 5 An *itemset* is a non-empty set of at most two items: a word and its corresponding POS tag; the *itemset* may only contain a word or a POS tag. A *sequence* is an ordered list of itemsets, where each itemset correspond to a word in the *candidate connector*. □

The candidate connector "including/VBG but/CC not/RB limited/VBN to/TO" can be written in the form of a sequence, i.e.,

$$< (including, VBG), (but, CC), (not, RB), (limited, VBN)(to, TO) >$$

in which $(including, VBG)$ is an itemset of two items "including" and "VBG".

Thus, we transform our problem of finding frequent substrings from candidate connectors into one of finding frequent subsequences among all these sequences, with the *constraints*: (i) a resulting frequent subsequence must have continuous itemsets, and (ii) each itemset in the subsequence should contain at least one item.

The item missing from an itemset is represented by a wildcard *. In the above sequence, $< (including, VBG), (*, CC) >$ is a subsequence, but $< (including, VBG), (not, RB) >$ is not because the itemsets are not continuous in the original sequence. Also we require a frequent sequence to be closed that no super-sequences of it will

have the same support. We did not adopt the maximal pattern strategy that will only retain the longest frequent sequence and abandon all its sub-sequences. For example, both

$$< (including, VBG), (but, CC), (not, RB), (limited, VBN), (to, TO)$$

and $< (including, VBG) >$ will be generated as closed pattern while the maximal pattern strategy will discard the latter one because it is the sub-sequence of the previous one. Here we gives a formal problem definition as below:

Definition 6 Given a *sequence database* containing all sequences representing all *candidate connector*s between two concepts of an instance, the *support* of a continuous subsequence is the number of sequences in the sequence database that contain it, while the itemsets in this subsequence are continuous in these original sequences. We want to find frequent closed continuous subsequence that its support is larger than pre-defined threshold and no super-sequence of it will have the same support. □

In order to find frequent closed sequential patterns with the constraints we defined, we tailored the BIDE+ algorithm [32] with respect to the generation of projected database, and pruning non-closed patterns.

The algorithm for this module is shown in Algorithm 3, in which we find patterns in different directions (parent in front of child or child in front of parent). In Algorithm 3, $constrainedBIDEplus$ is the BIDE+ algorithm as adapted to our constraints.

A closed frequent subsequence discovered this way will be used as a *generic pattern*: $NPList_1$ $Connector$ $NPList_2$, in which the $NPList$ can be a list of Noun Phrases or a single Noun Phrase, and the $Connector$ can be further represented as $constituent_1, ..., constituent_m$ where $constituent_i$ is $word_i/POS_i$, $*/POS_i$ or $word_i/*$ in which $*$ is a wildcard that matches with any word having that POS tag or any POS tag applied to the word respectively. A pattern will also determine which side on the connector is the parent and therefore the other side is the child.

However, we do not want the closed frequent subsequences to have too much overlap in terms of support in the hope of finding more distinct patterns. For

Algorithm 3 Frequent_Pattern_Discovery

Require: A set of instances (*SeedInstances*) of the form (*ParentNP*,*ChildNP*) where ISA or HASA relationship may hold between *ParentNP* and *ChildNP*

Require: A limit (*WindowSize*) on the number of words in the text window considered in each instance's occurrence

Ensure: A set of generic patterns (*CandidatePatterns*).

1: $Seqs = \varnothing$
2: $ReversedSeqs = \varnothing$
3: **for all** $instance \in SeedInstances$ **do**
4: **if** $instance = (ParentNP, ChildNP)$ **then**
5: add to *Seqs* tagged substrings w/ length $< WindowSize$ that occur between *ParentNP* and *ChildNP*
6: **end if**
7: **if** $instance = (ChildNP, ParentNP)$ **then**
8: add to *ReversedSeqs* tagged substrings with length $< WindowSize$ that occur between *ChildNP* and *ParentNP*
9: **end if**
10: **end for**
11: $Patterns = \varnothing$
12: $ReversedPatterns = \varnothing$
13: $Patterns \leftarrow constrainedBIDEplus(Seqs)$
14: $ReversedPatterns \leftarrow constrainedBIDEplus(ReversedSeqs)$
15: $CandidatePatterns \leftarrow Patterns \cup ReversedPatterns$

example, both $< (such, JJ), (as, IN) >$ and $< (such, JJ), (*, IN) >$ are generated as closed frequent pattern because they have different support, but most of time $< (such, JJ), (*, IN) >$ occurs it is in the form of $< (such, JJ), (as, IN) >$. So we propose a pattern generalization as post-processing that will discard a pattern p_1 like $< (such, JJ), (as, IN) >$ when there exists a sub-sequence p_2 with the same number of itemsets like $< such, JJ >, < *, IN >$ where

$$support(p_1)/support(p_2) \geq overlapRatio$$

in which *overlapRatio* is a threshold for the ratio determined empirically. We are conservative on this generalization so that we only consider sequences having the same length (the number of itemsets) because we still want to get patterns specific enough (relative longer and having fewer wildcards), which has higher precision.

Algorithm 4 describes the process.

Algorithm 4 Pattern_Generalization

Require: (*ClosedPatterns*) is a set of frequent closed patterns output by Algorithm 3

Require: (*OverlapRatio* is a pre-defined threshold for the ratio of a pattern's support to another pattern's support)

Ensure: (*GeneralizedPatterns*) is a set of frequent closed patterns

1: $FilteredPatterns = \varnothing$
2: **for all** Pattern $P \in ClosedPatterns$ **do**
3: **for all** Pattern $Q \in ClosedPatterns \wedge length(P) = length(Q) \wedge Q$ is a subsequence of P **do**
4: **if** $support(P)/support(Q) \geq OverlapRatio$ **then**
5: $FilteredPatterns \leftarrow Q$
6: **end if**
7: **end for**
8: **end for**
9: $GeneralizedPatterns \leftarrow ClosedPatterns - FilteredPatterns$

The closed frequent subsequences that are output and generalized are treated as candidate lexico-syntactic patterns. We make use of a scoring mechanism, described in the next section, for choosing the top patterns as the seed patterns for the next iteration of extraction of instance pairs.

4.4 Scoring of Patterns and SCIPs

We need a scoring mechanism to select seed SCIPS and seed patterns to identify new patterns and new concept pairs respectively, and decide the stopping criteria for the iterative process. It is prohibitively expensive to evaluate the actual precision of patterns and SCIPS at run-time.

In order to estimate the confidence of a pattern or a SCIP, we need to capture the association between a SCIP and any pattern contributing to its extraction in the pattern instantiation step. Similarly, we need to capture the association between a pattern and any SCIP contributing to its discovery in the frequent pattern discovery step. We follow the Point-wise Mutual Information (PMI) [8] framework in Pantel and Pennacchiotti [21] for scoring pattern and instances. Point-wise mutual

information measures the association strength between two events x and y, and is defined as:

$$pmi(x,y) = \log \frac{P(x,y)}{P(x)P(y)} \tag{4.2}$$

ESPRESSO estimates the PMI between an instance pair $i = (NP_1, NP_2)$ and pattern p as:

$$pmi(i,p) = \log \frac{|NP_1, p, NP_2|}{|NP_1, *, NP_2||*, p, *|} \tag{4.3}$$

In this formula, the numerator is the co-occurrence frequency of a pattern and an instance pair and the denominator is the product of their respective frequencies. Following Pantel and Ravichandran [22] a discount factor is applied so that this value is not biased too much by infrequent events. Then they define the reliability of an instance i:

$$r(i) = \frac{\sum_{p \in P'} \frac{pmi(i,p)}{max_{pmi}} * r(p)}{|P'|} \tag{4.4}$$

in which P' is the set of patterns in this iteration, and max_{pmi} is the maximum PMI value between any (pattern, instance) pair in this iteration. Symmetrically, the reliability of a pattern p, $r(p)$, is calculated as

$$r(p) = \frac{\sum_{i \in I'} \frac{pmi(i,p)}{max_{pmi}} * r(i)}{|I'|} \tag{4.5}$$

where I' is the set of instances used to find new patterns.

However according to the PMI formula listed in the ESPRESSO paper [21], the PMI value will always be negative, which contradicts the intuition that both instance score $r(i)$ and pattern score $r(p)$ are larger if the corresponding PMI value is larger. So instead of using this suspicious one, we follow the original definition of PMI in Formula Equation 4.2 that uses probability instead of frequency so that our PMI value is:

$$pmi(i,p) = \log \frac{\frac{|NP_1, p, NP_2|}{\sum_{\hat{p} \in P', \hat{i} \in I'} |NP_{1_{\hat{i}}}, \hat{p}, NP_{2_{\hat{i}}}|}}{\frac{|NP_1, *, NP_2|}{\sum_{\hat{i} \in I'} |NP_{1_{\hat{i}}}, *, NP_{2_{\hat{i}}}|} \frac{|*, p, *|}{\sum_{\hat{p} in P'} |*, \hat{p}, *|}} \tag{4.6}$$

In the above equation, we divide the frequency value in the numerator and the denominator with corresponding sum values, namely the sum of co-occurrence frequency for all pairs of instance and pattern, the sum of frequencies of all in-

stances, and the sum of frequencies of all patterns respectively. Here, $\hat{i}$ ranges over instances, i.e., $\hat{i} = (NP_{1_{\hat{i}}}, NP_{2_{\hat{i}}})$.

Whereas ESPRESSO uses seed instances, we start with pre-defined patterns. In the first iteration of pattern instantiation only, we estimated the precision of patterns by manual validation on a sampled output, and used those estimates as as initial scores.

The algorithm runs until no more new SCIPS can be found or the average score of patterns produced in this iteration is smaller than 50% of the average score of patterns from the previous iteration.

Chapter 5

Experiments and Evaluation

5.1 Dataset Description and Preprocessing

We evaluated our results on the following four datasets:

AEC: The Architecture, Engineering, and Construction dataset consists of the text data used by the construction firm in the process of constructing the Centre for Interactive Research on Sustainability (CIRS) building at the University of British Columbia. It is a web archive containing scheduling data, 3D design data, meeting notes, and reports. We extracted the text and applied basic cleaning. The resulting small corpus contains 18,805 sentences and 312,936 words. This fairly small dataset shows challenges for ontology extraction when data is sparse.

LP: LP[1] consists of text from *http://www.lonelyplanet.com*. This tourism-domain small-scale dataset has 18,950 sentences and 453,299 words; LP is also used by Cimiano and Staab [6].

MED: OHSUMED (or "MED" for short) consists of 348,566 medical references from MEDLINE[2] [15]. We use a large subset of this collection consisting of 1,221,462 sentences and 32,524,017 words. MED is a standard corpus in information retrieval.

[1] http://olc.ijs.si/lpReadme.html
[2] http://www.ncbi.nlm.nih.gov/pubmed/

Blue: The Bluestream collection[3] is a set of instructional manuals for a software product called XDocs, which is a component content management system developed by the Bluestream company[4] based in Vancouver. This corpus contains 4,295 sentences (81,087 words) after extracting text from about 300 XML files. It is the smallest dataset in this experiment and falls into Computer Science domain.

We did the following preprocessing steps on all datasets:

- We cleaned the data (e.g., we removed running footers).
- We broke the text into sentences with the LingPipe toolkit[5].
- We used the Stanford NLP tools to tokenize, tag, and parse the data.
- We built an inverted index with Lucene[6].

5.2 Competing Algorithms and Parameters

We compare LASER with two other algorithms:

ESPRESSO: ESPRESSO [21], discussed in detail throughout the paper, is our closest competitor.

GHC: GHC [6] (Chapter 2) did not describe how to generate the terms to build the taxonomy on. To overcome this, we used the C/NC-value method [11] proposed by the OntoGain system [10] to create multi-word candidate terms for GHC. We slightly modified C/NC to produce single-word terms in addition to multi-word terms since GHC performs poorly when only multi-word terms are input.

Both LASER and ESPRESSO [21] are iterative. LASER starts with the seed patterns in Chapter 3, and ESPRESSO is given 50 seed SCIPS for AEC, LP and Bluestrem and 100 seed SCIPS for MED. Since there exist no validated SCIPS for

[3] http://kb.bluestream.com/
[4] http://www.bluestream.com
[5] http://alias-i.com/lingpipe/
[6] http://lucene.apache.org/

both corpora, we randomly selected a set of SCIPS that have been labeled as valid when we evaluated the experiment result for LASER.

During the ISA/HASA Pattern Instantiation iteration, both algorithms pick the top k extracted (extended) SCIPS as seed SCIPS. For the small datasets like AEC, LP and Bluestream, we use all extracted (extended) SCIPS as seed SCIPS ; and k is set to 1500 on MED.

In the Frequent Pattern Discovery iteration, LASER chooses the top m patterns. ESPRESSO produces the top m patterns during the first run and generates patterns that increase in size by one pattern per round, e.g., the second round will find $m+1$ patterns. LASER sets m to 10 when finding ISA relationships on AEC and 5 for all other cases, and ESPRESSO has $m = 5$.

The only parameter for GHC is the number of input terms, n. The n most important terms from corpus chosen by our implementation of the C/NC-value method[11] are given as input, and GHC tries to find ISA relationships among them and build a hierarchy. We set $n = 400$ for Bluestream, $n = 1000$ for AEC, $n = 2000$ for LP and $n = 5000$ for MED because we want to keep the output size of different algorithms comparable.

Both GHC and ESPRESSO originally used Google[7] as an external source of ISA/HASA relationship evidence. We only implemented the web extension part in ESPRESSO with Microsoft Bing search API[8], for the following reasons:

1. Search engine service providers such as Google and Yahoo! have recently begun restricting the use of their search service API. Search service is not a free and abundant resource for experimental or academic use anymore.

2. GHC used web expansion to improve precision, but found that web expansion only improved precision by 2-3%.

We applied the three algorithms to the datasets in Section 5.1. LASER and ESPRESSO find ISA/HASA relationships while GHC is only able to produce ISA relationships.

In addition to measuring the algorithms' *precision* (i.e., what fraction of the results that are returned are correct), we would like to measure recall (i.e., what

[7] http://code.google.com/apis/websearch/

[8] http://www.bing.com/toolbox/bingdeveloper/

fraction of the correct results are returned). However, given that it is infeasible to fully find all ontological relationships in a large text repository, we measured *relative recall* — the number of valid relationships found by the algorithm divided by the total number of valid relationships found by all algorithms [20]. This allows us to also define *relative F-score* by replacing *recall* with *relative recall*. Thus

$$relative\ Fscore_{\beta} = (1+\beta^2) * \frac{precision * relative\ recall}{\beta^2 * precision + relative\ recall} \tag{5.1}$$

in which recall is weighted β times as important as precision. Therefore, F_1 weights precision and recall equally, $F_{0.5}$ weights precision as 2 times more important than recall, and F_2 weights recall as 2 times more important. We use these F-scores here because precision and recall may be weighted differently in different applications.

5.3 Comparison of ISA Results

Table 5.1: Precision and Total Number of ISA Results

System	AEC Precision	AEC Total	LP Precision	LP Total	MED Precision	MED Total	Blue Precision	Blue Total
LASER1	0.593	617	0.63	2198	0.6	19338	0.682	192
LASER1 HW	0.564	1070	0.5	3995	**0.61**	34390	0.634	344
LASER2	0.453	64	0.644	104	0.37	5323	0.657	35
LASER2 HW	0.459	111	0.642	179	0.42	9674	0.672	61
LASER	0.58	681	0.61	2302	0.55	24661	0.678	227
LASER HW	0.555	**1181**	0.6	**4174**	0.56	**44064**	0.640	405
ESPRESSO	0.673	55	**0.766**	141	0.53	3472	**0.736**	53
ESPRESSO HW	**0.674**	95	0.755	229	0.59	5814	0.721	86
ESPRESSO HW+W	0.337/0.562	406	0.59/0.68	1396	0.43/0.5	14077	0.322/0.439	**444**
GHC	0.337	734	0.51	1074	0.59	3557	0.407	302

Table 5.2: Relative Recall and F-score of ISA

	AEC				LP				MED				Blue			
System	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2
LASER	0.39	0.47	0.53	0.42	0.36	0.45	0.53	0.39	0.41	0.47	0.52	0.44	0.30	0.42	0.54	0.34
LASER HW	**0.65**	**0.60**	**0.57**	**0.63**	**0.65**	**0.62**	**0.61**	**0.64**	**0.75**	**0.64**	**0.59**	**0.70**	**0.51**	**0.57**	**0.61**	**0.53**
ESPRESSO	0.04	0.07	0.15	0.05	0.03	0.05	0.12	0.04	0.06	0.10	0.20	0.07	0.08	0.14	0.27	0.09
ESPRESSO HW	0.06	0.12	0.23	0.08	0.05	0.08	0.18	0.06	0.10	0.18	0.31	0.13	0.12	0.21	0.36	0.15
ESPRESSO HW+W	0.14	0.19	0.26	0.15	0.21	0.31	0.44	0.24	0.18	0.26	0.34	0.21	0.28	0.30	0.31	0.29
GHC	0.25	0.28	0.31	0.26	0.14	0.22	0.34	0.17	0.06	0.12	0.22	0.08	0.24	0.30	0.36	0.26

Using the stopping criteria in Section 4.4, LASER ran two ISA Pattern Instantiation iterations on all datasets. ESPRESSO only ran one ISA Pattern Instantiation iteration before it reached its stopping criteria.

Table 5.1[9] and Figure 5.1 show the total number of all output ISA relationships for each algorithm and the corresponding precision on all four corpora. We manually validated all relationships produced for AEC and Blue. For the other two corpora, we validated random 100 results if there were more than 1,000 relationships, otherwise did complete validation. LASER1 and LASER2 represent the relationships directly extracted from patterns during iteration 1 and 2; LASER is the total result from all iterations. HW denotes the results containing extended relationships found by the SCIP Extension step (Section 4.2). HW+W represents the result with both head word extension and web extension.

From this table we can see that ESPRESSO achieves the best precision on two datasets including AEC and LP and LASER achieves the best precision on MED and Blue. LASER dominates on three datasets on number of results returned. Since LASER's precision is comparable to ESPRESSO, this means LASER extracts many more valid relationships than ESPRESSO. Head word extension increases the number of relationships found by both LASER and ESPRESSO, with precision remaining about the same or decreasing a little bit because of errors in finding head words. ESPRESSO's web expansion produces many additional relationships, but it markedly degrades precision.

There are two numbers in each precision column of ESPRESSO HW+W; the first measures precision on relationships found in the domain. The second measures precision if the relationship is valid in *any* domain. For example, ISA(accessories, necklace) is extracted by ESPRESSO on the AEC dataset. This is not valid in the architecture domain because necklace is not a concept in this domain — in this domain, accessories stands for construction or mechanical equipment.

GHC relies heavily on ISA relationships between a term and its head word, e.g., ISA(system, heat recovery system), which are fairly trivial. Neither LASER, nor ESPRESSO output these relationships. The input terms extracted for MED contain a higher percentage of multi-word terms (32%) than those of AEC (23%), so

[9] In each table, the "best" result per column is bolded.

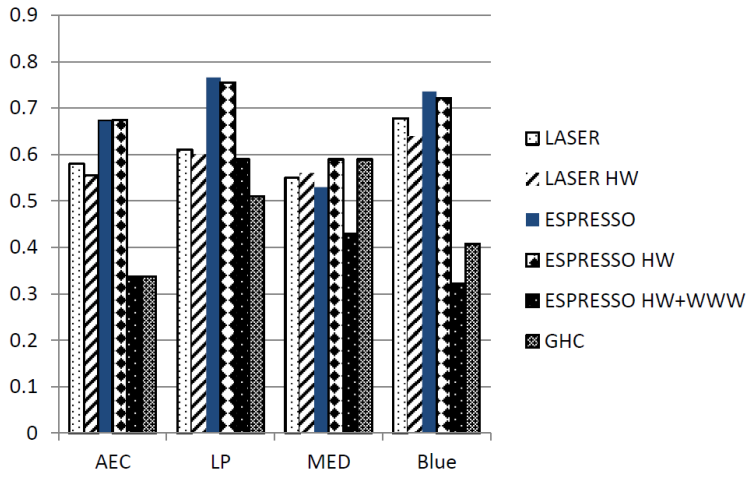

Figure 5.1: Precision Result for ISA

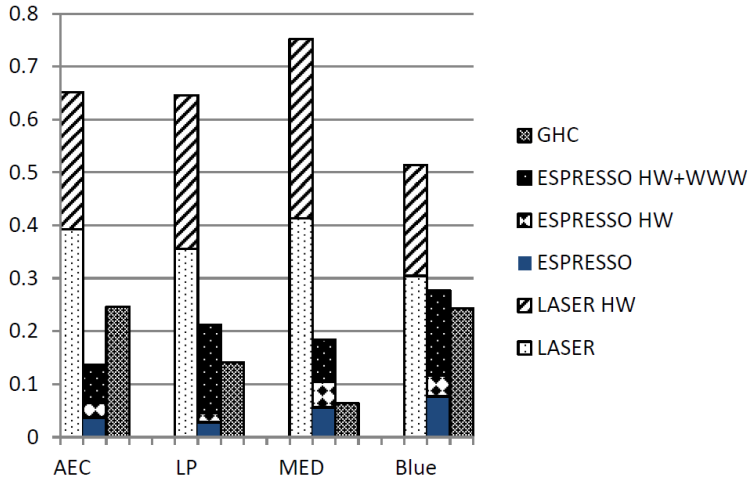

Figure 5.2: Relative Recall Result for ISA

GHC performs much better on the MED corpus: more "trivial" relationships can be found.

Table 5.2 gives the relative recall and different F-scores for algorithms on these datasets. Testing the validity of all relationships from the two larger datasets is

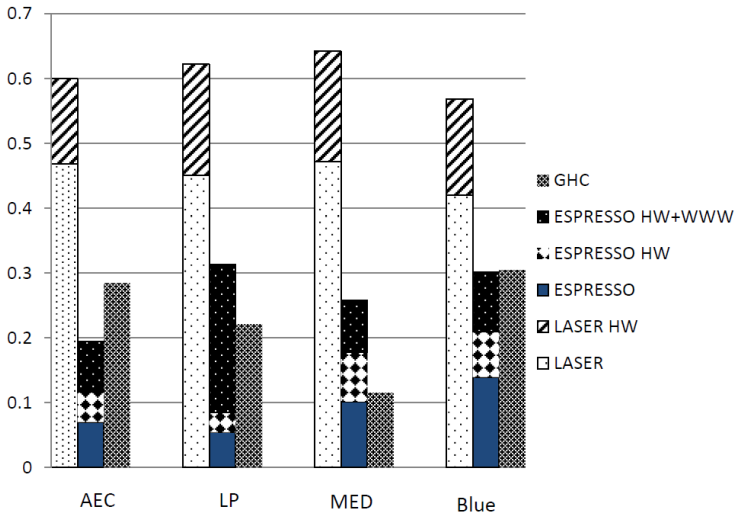

Figure 5.3: Relative F-score Result for ISA

impractical, so for those, we estimate *relative recall* by:

$$relative\ recall \approx \frac{precision * |SCIPs|}{\sum precision_x * |SCIPs|_x} \tag{5.2}$$

in which the number of valid relationships produced by an algorithm is estimated by the product of sample precision and the number of all generated SCIPS. Summing the estimated valid relationships for all competing algorithms, yields the number of all valid relationships from all systems' output, which is an overestimate of the real value. Therefore, the estimated relative recall is an under estimate but still reflects the difference between systems.

LASER HW outperforms the other two algorithms and corresponding extensions in terms of relative recall and F-scores, thanks to the large output and stable precision. In contrast, ESPRESSO suffers from low relative recall. This behavior is consistent on both small and large datasets, which reflects a problem of starting an iterative algorithm from seed SCIPS. Although ESPRESSO's set of SCIPS are valid, the distribution of these seeds in the corpus is unknown beforehand, leading to possibly re-discoverying the same pattern repeatedly and hence a consistently low recall. GHC has better relative recall and F-score than ESPRESSO on AEC and Blue

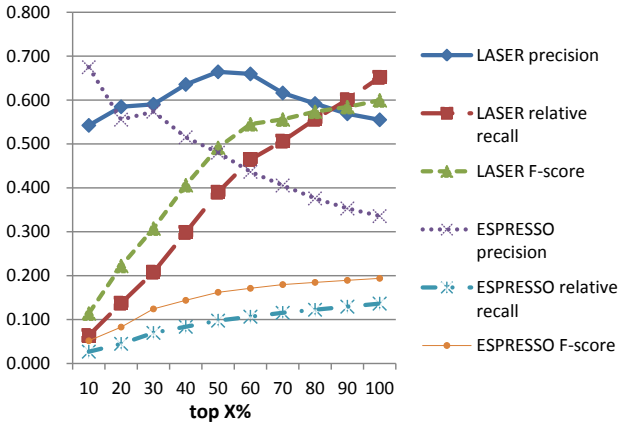

Figure 5.4: Comparing the Top X% ISA SCIPS

even when its precision is low on these two datasets. On the large corpus, GHC has worse relative recall mainly because the agglomerative clustering algorithm does not scale well. As we show later, even running GHC with 5000 terms took more than two days.

Figure 5.2 shows the relative recall of ISA relationships. Since the variation in precision across different parts of the algorithms (e.g., the difference between ESPRESSO and ESPRESSO HW) is relatively low, the F-scores (Figure 5.3) are very similar to the relative recall graph. We can conclude that Head Word Extension and Web Extension both improve F-scores, and LASER HW dominates consistently.

It is interesting to see how these measures vary for output SCIPS with different scores. In Figure 5.4 we plot precision, relative recall, relative F1-Score as a function of X, in which top X% stands for the top X% SCIPS having highest scores on AEC. The precision of LASER remains relatively steady even when we dig to the bottom of the scores, i.e., as X increases. Additionally, both LASER's relative recall and F-score increase rapidly as X increases, showing LASER's dominance over the competition.

Table 5.3: Precision and Total Number of HASA Results

System	AEC Precision	AEC Total	LP Precision	LP Total	MED Precision	MED Total
LASER1	0.415	82	**0.626**	673	**0.42**	4011
LASER2	0	0	0.429	7	0.25	417
LASER	**0.415**	**82**	0.624	**680**	0.39	**4428**
ESPRESSO	0.25	4	0.588	51	0.35	428
ESPRESSO HW	0.11	9	0.521	71	0.31	649
ESPRESSO HW+W	0.1	10	0.454/0.471	121	0.34/0.39	1072

5.4 Comparison of HASA Results

LASER ran one HASA Pattern Instantiation iteration on AEC and two iterations on other two larger datasets LP and MED. ESPRESSO still ran only one HASA Pattern Instantiation iteration. Table 5.3 shows that both the precision and number of HASA relationships are worse than ISA relationships for all algorithms. This is because in a corpus, HASA relationships are not as frequent as ISA relationships. We did not show the result for the Blue dataset because ESPRESSO fails to produce any HASA relationships on this extremely small corpus.

LASER outperforms ESPRESSO in every case for all datasets. One thing to note is that LASER only extends ISA SCIPs in the SCIP Extension step (Section 4.2), but ESPRESSO extends both ISA and HASA SCIPs. We made this choice because HASA has different semantic meanings from ISA and contains many subtypes [12]. For example, HASA(treatment of occlusive disease, endarterectomy) is a valid relationship from MED, but its head word extension HASA(treatment, endarterectomy) does not make sense because "treatment" is too abstract that "endarterectomy" is not part of "treatment" in the general sense. ESPRESSO's drop in precision when it applies HASA headword extension also reflects this.

The relative recall and F-scores on the three corpora are presented in Table 5.4. LASER dominates both measurements consistently while ESPRESSO still suffers from low recall. Figure 5.5, 5.6, and 5.7 show that the various components contribute at about the same fashion as they do for ISA relationships.

Similar to the ISA experiments, we plot precision, relative recall, relative F1-Score as a function of X, in which top X% stands for the top X% SCIPs having highest scores on LP. We choose LP instead of AEC because the number of valid

Table 5.4: Relative Recall and F-score of HASA

	AEC				LP				MED			
System	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2
LASER	**0.97**	**0.58**	**0.47**	**0.77**	**0.89**	**0.73**	**0.66**	**0.82**	**0.83**	**0.53**	**0.44**	**0.68**
ESPRESSO	0.03	0.05	0.10	0.04	0.06	0.11	0.22	0.08	0.07	0.12	0.20	0.09
ESPRESSO HW	0.03	0.05	0.07	0.03	0.08	0.13	0.24	0.09	0.10	0.15	0.22	0.11
ESPRESSO HW+W	0.03	0.05	0.07	0.03	0.12	0.18	0.29	0.14	0.17	0.23	0.29	0.19

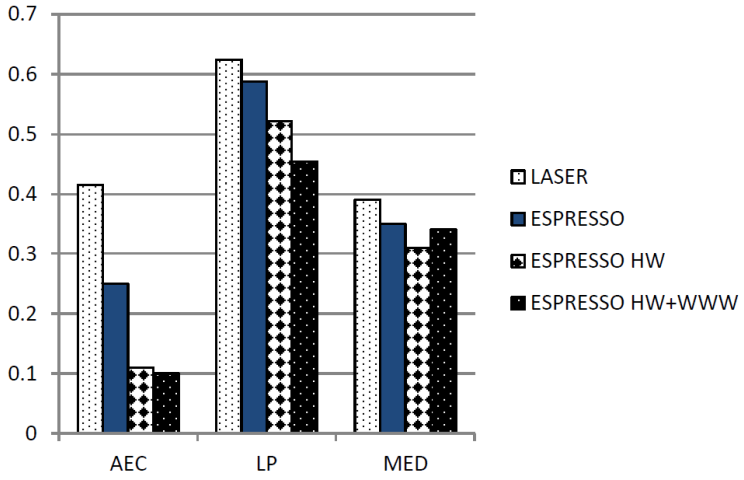

Figure 5.5: Precision Result for HASA

HASA relationships found by ESPRESSO is too small. The curve (Figure 5.8) looks consistent with the ISA result, showing LASER is stable algorithm.

5.5 Comparison of Running Time

Table 5.5 shows the running times for extracting ISA relationships. LASER and LASER HW have the same running time because both versions of LASER require headword extension for seed generation; the only difference is whether we count these extended relationships during evaluation. This is also true for ESPRESSO and ESPRESSO HW.

LASER is the most efficient algorithm and is between 1.4 times and two orders of magnitude faster than other algorithms. Indeed LASER's constrained closed se-

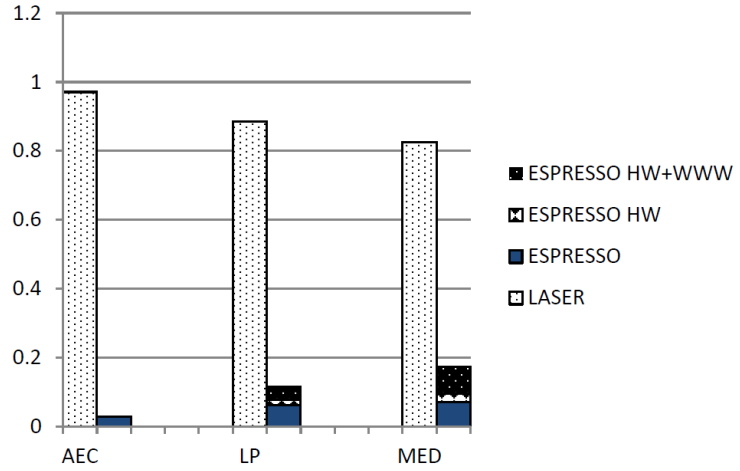

Figure 5.6: Relative Recall Result for HASA

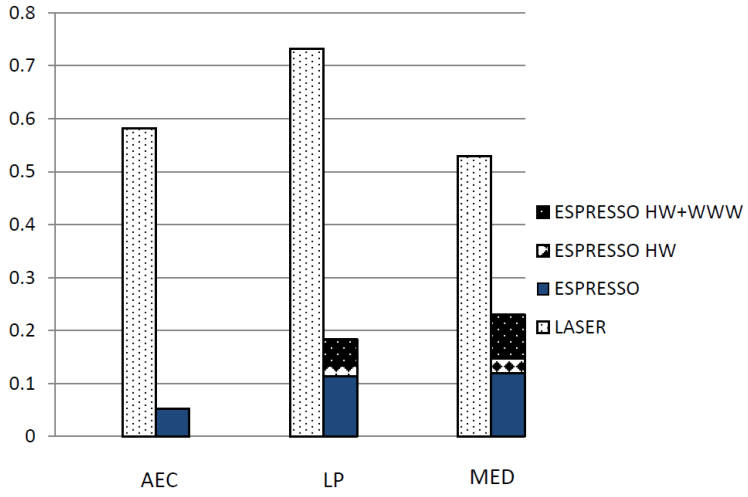

Figure 5.7: Relative F-score Result for HASA

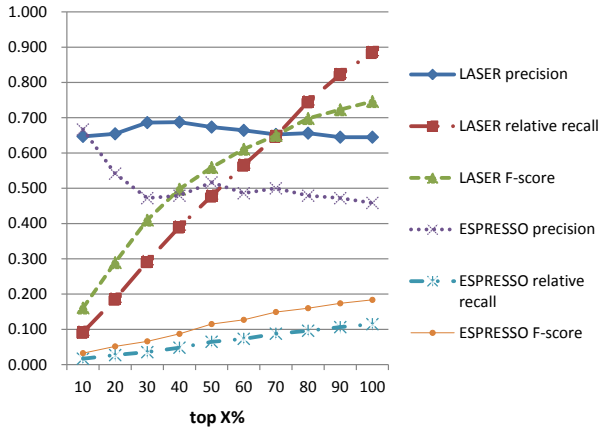

Figure 5.8: Comparing the Top X% HASA SCIPS

Table 5.5: ISA Extraction Running Time (in Seconds)

System	AEC	LP	MED	Bluestream
LASER	65	107	6,862	27
ESPRESSO	518	308	9,627	133
ESPRESSO HW+W	10,939	10,439	72,154	2,727
GHC	1,160	1,709	>2 days	146

quential pattern mining approach to finding new patterns is much more efficient than ESPRESSO's frequent substring finding using a suffix tree. ESPRESSO HW+W takes even longer because search engines constrain the frequencies of queries. This can only get worse as more search engines limit their access. The running time for GHC is quadratic in the number of input terms because agglomerative clustering requires pairwise term similarity. This becomes GHC's bottleneck when the number of input terms gets larger. Indeed, it takes GHC more than two days to finish on an input of 5,000 terms! The running time for extracting HASA relationships is similar to the ISA case and it is shown in Table 5.6.

In summary, LASER (equals or) outperforms the two other algorithms on precision, relative recall and F-score for both ISA and HASA relationships in most cases.

Table 5.6: HASA Extraction Running Time (in Seconds)

System	AEC	LP	MED
LASER	49	22	2,302
ESPRESSO	149	92	17,705
ESPRESSO HW+W	178	496	22,506

While ESPRESSO suffers from low recall and GHC finds too many "trivial" relationships, LASER outputs a lot of relationships on both small and large corpora, which shows the superiority of using an iterative framework that starts from reliable seed patterns. Using parse tree information and identifying appropriate noun phrases from nested noun phrases, contribute to the discovery of more complex and accurate relationships. This parse-once-use-many-times strategy and the adaptation of constrained frequent closed sequential pattern mining make LASER very efficient, while the competing algorithms have serious running time bottlenecks.

Chapter 6

Conclusions and Future Work

Many state-of-the-art algorithms for learning ontologies from free text confine themselves to concepts represented as single-word terms or common compounds. In contrast, we find a richer ontology by covering multi-word terms. We build on and extend previous pattern-based iterative frameworks [14, 21], and make the following contributions:

1. We identify concepts in ISA/HASA relationships by analyzing parse trees instead of simple POS tag matching, and use an efficient parse-once-use-many-times strategy.

2. We develop a novel algorithm to determine the appropriate noun phrases from nested noun phrases present in the corpus.

3. We tailor sequential pattern mining to find constrained frequent patterns consiting of words, POS tags, and wildcards.

We empirically show on four real web datasets that LASER performs very well and is particularly good at stably extracting rich and complex concepts and ISA/HASA relationships between them, regardless of the size of corpus or data sparsity. In terms of precision, it is comparable to or better than the competitors while in terms of relative recall and F-scores it significantly and consistently outperforms them. It is also much more efficient than the competing algorithms on running time and is scalable to very large data sets.

An interesting future challenge is to post-process concepts found by LASER with statistical methods to boost the precision even further while maintaining scalability.

Bibliography

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ICSW*, 2007. → pages 8

[2] M. Berland and E. Charniak. Finding parts in very large corpora. In *ACL*, pages 57–64. ACL, 1999. → pages 3, 13

[3] C. Biemann. Ontology learning from text: A survey of methods. *LDV Forum*, 20(2):75–93, 2005. → pages 1, 5

[4] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB*, 1999. → pages 3

[5] S. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *ACL*, 1999. → pages 1, 6

[6] P. Cimiano and S. Staab. Learning concept hierarchies from text with a guided hierarchical clustering algorithm. In *ICML workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, 2005. → pages 6, 7, 13, 15, 30, 31

[7] P. Cimiano and J. Völker. Text2onto. *Natural Language Processing and Information Systems*, pages 227–238, 2005. → pages 8

[8] T. Cover and J. Thomas. *Elements of information theory*, volume 6. Wiley Online Library, 1991. → pages 27

[9] P. Derose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. In *VLDB*, 2007. → pages 8

[10] E. Drymonas, K. Zervanou, and E. Petrakis. Unsupervised ontology acquisition from plain texts: the OntoGain system. *Natural Language Processing and Information Systems*, pages 277–287, 2010. → pages 6, 31

[11] K. Frantzi, S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000. → pages 31, 32

[12] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135, 2006. ISSN 0891-2017. → pages 3, 13, 40

[13] Z. Harris. Distributional structure. *Word*, 1954. → pages 5

[14] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*. ACL, 1992. → pages 1, 3, 6, 10, 13, 45

[15] W. Hersh, C. Buckley, T. Leone, and D. Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR*, 1994. → pages 30

[16] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *Arxiv preprint cmp-lg/9709008*, 1997. → pages 19

[17] D. Lin. An information-theoretic definition of similarity. In *ICML*, volume 1, pages 296–304, 1998. → pages 19

[18] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *SEKE*, pages 231–239, 2000. → pages 8

[19] C. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 59. MIT Press, 1999. → pages 11

[20] A. Moosavi, T. Li, L. Lakshmanan, and R. Pottinger. Ontectas: Bridging the gap between collaborative tagging systems and structured data. In *CAiSE*, 2011. → pages 6, 33

[21] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *COLING*, pages 113–120. ACL, 2006. → pages 1, 3, 7, 15, 22, 27, 28, 31, 45

[22] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In *HLT/NAACL*, 2004. → pages 6, 28

[23] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet:: Similarity: measuring the relatedness of concepts. In *HLT/NAACL*, pages 38–41, 2004. → pages 19

[24] H. Poon and P. Domingos. Unsupervised ontology induction from text. In *ACL*, pages 296–305, 2010. → pages 8

[25] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. An algebraic approach to rule-based information extraction. In *ICDE*, 2008. → pages 8

[26] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11:95–130, 1999. → pages 19

[27] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *SIGIR*, pages 206–213, 1999. ISBN 1581130961. → pages 1, 6, 7

[28] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*, page 7, 1997. → pages 8

[29] Srikant and Agrawal. Mining sequential patterns: Generalizations and performance improve. *EDBT*, 1996. → pages 24

[30] F. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007. → pages 8

[31] F. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW*, 2009. → pages 8

[32] J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. 2004. ISSN 1063-6382. → pages 25

[33] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *CIKM*, pages 41–50, 2007. → pages 8

[34] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW*, pages 635–644, 2008. → pages 8

[35] E. Zavitsanos, G. Paliouras, G. Vouros, and S. Petridis. Learning subsumption hierarchies of ontology concepts from texts. *WIAS*, 8(1):37–51, 2010. → pages 8