# AN EMPIRICAL ANALYSIS OF LEXICAL POLARITY AND CONTEXTUAL VALENCE SHIFTERS FOR OPINION CLASSIFICATION 

by<br>Adam Longton<br>B.Sc., University of British Columbia, 2003

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)

October 2008
© Adam Longton, 2008


#### Abstract

This work is concerned with the automatic understanding of evaluative text. We investigate sentence level opinion polarity prediction by assigning lexical polarities and deriving sentence polarity from these with the use of contextual valence shifters. A methodology for iterative failure analysis is developed and used to refine our lexicon and identify new contextual shifters. Algorithms are presented that employ these new shifters to improve sentence polarity prediction accuracy beyond that of a state-of-the-art existing algorithm in the domain of consumer product reviews. We then apply the best configuration of our algorithm to the domain of movie reviews.


## Table of Contents

Abstract ..... ii
Table of Contents ..... iii
List of Tables ..... v
List of Figures ..... vi
Acknowledgements ..... vii
1 Introduction .....  1
2 Literature Review ..... 5
2.1 Subjective Language Classification ..... 5
2.2 Opinion Analysis ..... 7
2.3 Evaluation and Appraisal in Linguistics ..... 8
3 Framework ..... 10
3.1 Overview ..... 10
3.2 Corpus Processing ..... 12
3.2.1 Part of Speech Tagging ..... 12
3.3 Baseline Algorithms ..... 13
3.4 Lexical Polarity and Lexicon Expansion ..... 14
3.4.1 WordNet ..... 14
3.4.2 Log Likelihood ..... 15
3.5 Contextual Valence Shifters ..... 15
3.6 Coverage and Accuracy ..... 16
4 Experiments and Analysis ..... 17
4.1 Lexicons and Corpora ..... 17
4.2 Iterative Failure Analysis ..... 18
4.3 Baseline Experiments ..... 20
4.3.1 Hu -Liu Algorithm ..... 20
4.3.2 Feature Based Effective Opinion ..... 21
4.3.3 Polysemy and Bipolarity in WordNet Expansion. ..... 23
4.4 Lexical Polarity Analysis ..... 24
4.4.1 Intelligent HM Lexicon ..... 24
4.4.2 Polarity in other Parts of Speech and the GI Lexicon ..... 26
4.4.3 Combined Lexicon ..... 27
4.4.4 Nonpolar and Bipolar Words and the Robust Lexicon ..... 29
4.4.5 The Augmented Lexicon ..... 30
4.5 Contextual Valence Shifters ..... 31
4.6 Remaining Language Features Affecting Polarity ..... 37
4.7 Default Scoring Algorithms ..... 38
4.8 Application to Another Domain: Movie Reviews ..... 39
4.9 Discussion of Results ..... 43
5 Conclusions and Further Work ..... 45
References ..... 48
Appendix - Project Code ..... 51

## List of Tables

Table 1: Hu-Liu Implementation ..... 21
Table 2: Hu Lexicon Analysis ..... 22
Table 3: HulexWN - Coverage for HL Corpus ..... 22
Table 4: HM Lexicon with WordNet Expansion ..... 25
Table 5: HM Lexicon without WordNet ..... 25
Table 6: HM - Coverage for HL Corpus ..... 25
Table 7: XGI Lexicon - Adjectives ..... 26
Table 8: XGI Lexicon - Other Parts of Speech ..... 27
Table 9: XGI Lexicon - All Parts of Speech. ..... 27
Table 10: XGI - Coverage for HL Corpus ..... 27
Table 11: Combined Lexicon HX ..... 28
Table 12: HX - Coverage for HL Corpus ..... 28
Table 13: Robust Lexicon myHX. ..... 29
Table 14: MyHX - Coverage for HL Corpus ..... 30
Table 15: Augmented Lexicon myHX + ..... 31
Table 16: MyHX+ - Coverage for HL Corpus ..... 31
Table 17: MyHX + with Negation Tuning ..... 32
Table 18: Augmented Lexicon MyHX+ with Shifters ..... 34
Table 19: MyHX+ with Shifters - Coverage for HL Corpus ..... 35
Table 20: HX and myHX with Shifters - Coverage for HL Corpus ..... 35
Table 21: Hu Liu Lexicon with WordNet on Pang Corpus ..... 39
Table 22: HM Lexicon with and without WordNet ..... 40
Table 23: Part of Speech Analysis in Combined and Robust Lexicons ..... 40
Table 24: Robust Lexicon with Negation Shifter ..... 41
Table 25: Augmented Lexicon on Pang ..... 41
Table 26: MyHX+ with Shifters ..... 42
Table 27: MyHX+ with Shifters and WordNet Expansion ..... 42
Table 28: MyHX+ with Shifters, WordNet, and Log-Likelihood Expansion ..... 42

## List of Figures

Figure 1: Opinion Classification System Data and Modules............................................. 11
Figure 2: Opinion Classification System Flow................................................................. 11

## Acknowledgements

I would like to thank Giuseppe Carenini for his continual guidance, encouragement, and inspiration. I would also like to thank Kevin Leyton-Brown for his insightful suggestions. Many thanks go to those who have helped me along the way, including David Kirkpatrick, Will Evans, Nando de Freitas, Patrice Belleville, Ian Cavers, and Gregor Kiczales. Finally I would like to thank my family for all their help and my wife Sarah for her endless support.

## 1 Introduction

An important problem in the field of natural language processing is the automatic understanding of evaluative text. By evaluative text we mean text in which the author is expressing an opinion or sentiment toward a topic. The internet contains a large volume of this kind of writing and in a given domain there is usually more information than an individual or party could possibly manually process. One example domain is that of consumer reviews. It is commonplace for businesses to provide a mechanism on their website for consumer feedback. Both the business and the consumers value the information found here for making decisions but because of the potentially large amount of information there is a need for fast interpretation and a summary of general trends in the opinions presented. Opinions about businesses and their products and services can also extend beyond their own websites, into blogs and other news media. This information in turn can influence investors and consumers. Reviews are important to many industries. Consider the entertainment industry where opinions about movies and music are highly influential. The service industry is another important domain where for example reviews of hotels and restaurants can be a determining factor in people's decision making. There is also value in the automatic analysis of public opinion from a political and governmental perspective. Detecting trends in opinions of policies and platforms could help shape the political and social landscape.

The goal of this work is to investigate current methods for discovering the polarity of people's opinion in text and to present various techniques for improving these methods. By polarity of opinion, we mean whether an author is expressing a positive or negative opinion toward a subject of discussion. This feature of text can be viewed from different levels of discourse. One might be interested in opinions found at the word, phrase, clause, sentence, document, or multi-document level. This depends on the application. However,
information from lower levels of discourse might be useful in making predictions about the opinions expressed at higher levels. Methods fall into two broad categories: the lexical and grammatical levels of analysis. The lexical task is to identify polarity information at the lexical or individual word item level. This is commonly done by starting with a lexicon ("seed set") of known polarity words, sometimes expanding this set with various techniques, and then assigning a polarity score to words in a test corpus using these polar words. Once the level of discourse has been chosen, the grammatical or contextual task is to compute a score for the items at that level using local lexical information and possibly other contextual information.

For example, say we are interested in sentence level opinions. We might assign the words in the sentence a score, such as +1 or -1 (the lexical task), and then add the word scores to obtain a sentence score (the contextual task). To improve results, we might consider the effects of grammatical constructions that affect polarity such as negation and modality. In this case, we must identify these grammatical features in the sentence and compute an adjusted sentence polarity score based on their effect on polarity information contained at the lexical level. For example, negation tends to invert the polarity that would otherwise be derived from the lexical entries alone. Determining exactly how various grammatical constructions can shift polarity is a task in itself.

Our current work is primarily concerned with sentence level polarity prediction. We perform this task by following the commonly applied unsupervised strategy of starting from a set of seed words of known polarity. Our distinguishing contributions include the validation of using polarity words with parts of speech beyond adjectives and the demonstration of iterative failure analysis to refine our seed set and prune bipolar words. Furthermore, we contribute to the classification of various contextual shifters that adjust predictions based on linguistic context. We feel that although the important applications include document classification (reviews, editorials, and so on) or document and multidocument opinion summarization, the atomic components of polarity are the lexical items, and a bottom-up approach to computing overall polarity is appropriate. We have chosen the sentence level because it is a convenient and relatively localized context that contains
both lexical items and grammatical constructions. We hypothesize that discoveries made at this level may contribute to tasks dealing with higher levels, or at least provide insight into the interaction between polarity-bearing components and polarity-shifting constructions in general.

This level of polarity prediction has also been looked at by Yu and Hatzivassiloglou [YH03] and Hu and Liu [HLO4]. However, the amount of linguistic knowledge that is employed in both cases is sparse. An important contribution of our work is the integration of knowledge drawn from linguistics to provide a more solid theoretical foundation.

We present various basic algorithms in Chapter 3, and then through an iterative process of failure analysis using a corpus of consumer product reviews, we develop context-based improvements to produce a more accurate algorithm.

To support the external validity of our findings, we have also applied the best configuration of our algorithm to another domain - that of movie reviews. We found that our algorithm performs significantly better than the baseline in this context.

To summarize the key contributions of this thesis:

- we validate the use of polarity words from parts of speech other than adjectives, including nouns, adverbs, and verbs, for sentence opinion classification
- we demonstrate the use of iterative failure analysis as a means to efficiently refine polarity word sets by eliminating words that are bipolar (both positive and negative in different contexts) or those which have nonpolar (neutral) senses
- we show the limitations of opinion classification that uses feature words for additional sentiment information
- we develop a set of contextual valence shifters that adjust sentence polarity using our test corpus as well as a methodology for discovering further shifters
- we provide a more solid theoretical basis for the use of shifter constructs in inferring overall sentence opinion by correlating our findings with linguistics literature


## 2 Literature Review

Recently there has been a growing body of work related to evaluative language processing. The broader area of research that includes this task is concerned with detecting and classifying subjective language. Subjective language, in contrast to objective or purely factual language, can be defined as that which contains evaluations, emotions, and other content that is not factually verifiable but is rather a form of personal expression. Although our concern is primarily with opinions, this work is relevant in that similar techniques can be found across the field and detecting subjective language more generally is needed if opinions are to be extracted from large collections of different kinds of text. Within the area of opinion classification specifically there is also a growing body of research. However, it tends to be light in the application of linguistic features and structure and focuses on more statistical and atomic properties of text. It is our intention to determine the limitations of these approaches and the point where richer linguistic features become necessary to understand opinion of text more accurately.

### 2.1 Subjective Language Classification

Work has been done by Wiebe et al. $[\mathrm{W}+99]$ to determine what the appropriate categories of subjectivity are and what linguistic features might be predictive of subjectivity. Machine learning has been applied to this task and the need to develop appropriate and accurate classifiers was shown by Wiebe et al. [W+99] and Riloff et al. [R+03]. To perform supervised learning experiments, text annotated for subjectivity is required. [W+99] have developed such corpora. These corpora constitute a first attempt at this
kind of annotation, and are relatively small, but larger corpora have also been recently developed by Wilson and Wiebe [WW03].

One of the first issues at hand is the level of classification. Classification can be done at the document level, where entire texts such as news articles are classified as subjective or objective (overall seeming to express opinions or just stating facts). This was investigated for instance in Wiebe et al. [W+01]. Alternatively, classification can be done at the sentence level as in [W+99], or within sentences, at the clause or even expression and individual word level, as in the work of Turney and Littman [TL03].

In addition to the level of classification, the categories of classification must also be decided upon. The simplest and coarsest category set is the binary classification subjective or objective. But because this can be difficult to decide for some sentences, or sometimes a finer level of detail is desired, other categorizations have been looked at. Yu and Hatzivassiloglou [YH03] look at the polarity of subjectivity, that is, whether an opinion is positive or negative in sentiment. Gordon et al. [G+03] looks at particular kinds of attitudes. [W+99] uses subjective and objective categories but with levels of certainty attached, on a scale from 1 to 4.

One of the first attempts to do classification at the sentence level was the work of Wiebe et al. [W+99]. They developed two disjoint, manually annotated corpora consisting of complete articles randomly selected from the Wall Street Journal Treebank Corpus. Four judges each independently tagged the non-compound sentences and conjuncts of compound sentences for whether they were subjective or not.

Once the corpora were tagged, machine learning experiments were performed. Probabilistic classifiers for subjectivity were used that were developed by Bruce and Wiebe [BW99]. These were based on the class of probability models known as decomposable models and were used to find text features that are probabilistically indicative of the subjective tag. Five part-of-speech features, two lexical features, and a paragraph feature were found and used in the experiments. They considered the naive

Bayes, full independence, and full interdependence probabilistic models as well as models generated from those using forward and backward search. The model then chosen was the one with the best accuracy for each training set.

Work has been since done to identify richer linguistic features to be used in classifiers. Wiebe [W00] uses clustering methods, and Riloff et al. [R+03] has looked at extraction patterns. Specific parts of speech have also been more carefully looked at in their usefulness for subjectivity tagging in [W00] and [R+03].

### 2.2 Opinion Analysis

There has also been work involving the task of predicting the polarity of opinions present in subjective language. One of the main contributions to this analysis is the work by Yu and Hatzivassiloglou [YH03]. This was motivated by earlier work done by Hatzivassiloglou and McKeown [HM97], where they took an initial seed set of positive and negative adjectives and then grew this list by looking at the participation of these words in conjunctions. If a known positive word occurs in an "and" with another word, then this word can be inferred to be positive too. This is the same for negative words. Similarly, if it occurs with "but" and another word, the other word has the opposite polarity.

Pang et al. $[\mathrm{P}+02]$ used different machine learning techniques to classify movie reviews as positive or negative in sentiment. They examine the Naïve Bayes, Maximum Entropy, and Support Vector Machines classifiers, and discuss differences between sentiment and topic classification. Turney and Littman [TL03] have also looked at review classification, but used web search hits to calculate polarity scores. They infer polarity from association with known polarity words using two measures of association - Pointwise Mutual Information and Latent Semantic Analysis. Although not concerned with opinions, Cilibrasi and Vitanyi [CV04] have recently developed the notion of Google distance to
measure semantic relationships between words. This could be useful in performing web based classification tasks like that of [TL03].

There has also been a line of research that involves opinion polarity detection that is specific to given subject targets. These could be the names of the company or product in question, the movie being reviewed, or a specific actor in a movie, depending on the desired task. Yi et al. $[\mathrm{Y}+04]$ have developed a system called Sentiment Analyzer that first extracts what it thinks are the relevant subject terms, and then assigns an opinion polarity to statements about them. This is done using a sentiment dictionary of words and phrases that have been labeled with a known sentiment value. A similar approach has been taken by Hu and Liu [HL04]. They give scores to sentences that include specific product features based on counting positive and negative constituent words. They also add the context sensitive step of inverting the polarity in the presence of negation words like 'not'.

### 2.3 Evaluation and Appraisal in Linguistics

In the field of linguistics, the use of language to express opinions is variously known as evaluation, authorial stance, and appraisal. Linguists organize types of evaluative language into hierarchies and categorize the terms and forms associated with different levels of appraisal into markers of stance. We are interested in modeling some of these markers of authorial stance and applying them to our automatic opinion classification scheme.

Martin's article in Text [M03] breaks down language used for appraisal into three classes: engagement, attitude, and graduation. Some of these contain features that can shift a lexical item's contribution of positive or negative opinion to the text. Engagement involves positioning one opinion in relation to another and includes features like projection, modality, polarity, and concession. These will evidently provide a context
within which a lexical item can be interpreted and allow for possible adjustments to the item's polarity contribution to the sentence. For example, projection might involve a report of a polarized statement but not be making such a statement. Concession might be similarly affirming some fact, like "He seemed angry", while modality might be acknowledging a possibility without committing to an opinion on it, in for example, "It might be really fun". Graduation involves gradability and includes features that raise or lower the degree of an evaluation, which can again shift the contribution.

The collection of linguistic papers compiled in the book Evaluation in Text: Authorial Stance and the Construction of Discourse, edited by Hunston and Thompson [HT00], contains discussions of contextual situations that convey or affect polarity. Biber and Finegan's stance markers are discussed in [TH00] and these include the following categories:

1. adverbs indicating affect, certainty, and doubt (e.g. definitely)
2. adjectives indicating affect, certainty, and doubt (e.g. happy)
3. verbs indicating affect, certainty, and doubt; (e.g. enjoy)
4. hedges (vague language e.g. about, sort of)
5. emphatics (e.g. for sure, really)
6. modals indicating possibility, necessity, and prediction (e.g. could be, should have been)

These too indicate features that have the ability to contribute or modify the opinion being expressed in evaluative text. Moon's Fixed Expressions and Idioms in English: A Corpus-Based Approach [RM98] also has a section on evaluation, and describes situations that cause a shift or "subversion" of evaluative orientation. We will refer to these when compiling lists of polarity sensitive linguistic patterns.

## 3 Framework

To achieve the sentence-level polarity prediction task, we have formulated a generalized algorithm that reflects the basic approach taken in the related work and have compiled a set of improvements to this algorithm that we investigate in Chapter 4. The set of improvements includes techniques developed by other researchers, modifications of existing techniques, and techniques we have invented.

### 3.1 Overview

The simplest baseline we can use for comparison is to take the polarity of the most frequent class (e.g., positive) as the polarity of all our test sentences. Thus any algorithm must at least achieve better accuracy than the frequency of the most frequent class. The common aspect of the various sentence polarity prediction techniques in the related work seems to be the use of a polarity lexicon. This is a precompiled list of positive and negative words. These words are either used to assign polarity scores to other words or to simply contribute to the score of a sentence. This is what we mean by a generalized algorithm. Beyond the most-frequent-class baseline, a simple algorithm is to compare the sets of positive and negative lexicon words found in a sentence and to take the class of the larger set as the prediction. This will be taken as the basic algorithm upon which various improvement techniques can be applied.

| Corpora | Part of Speech Taggers | Lexicons | Lexicon Expansion Algorithms | Contextual Valence Shifter Modules |
| :---: | :---: | :---: | :---: | :---: |
| - Hu-Liu <br>  Product <br>  Reviews <br> - Pang-Lee <br> Movie  <br>  Reviews | - NLProcessor <br> - Stanford Tagger | - HatzivassiloglouMcKeown <br> - General Inquirer <br> - Hu-Liu Seed Set <br> - Longton-Carenini <br> Refined Lexicons | - WordNet <br> - LogLikelihood | - Hu-Liu <br> Algorithm <br> - Longton-Carenini Algorithm |

Figure 1: Opinion Classification System Data and Modules
Our framework consists of the following components: corpora, part of speech taggers, lexicons, lexicon expansion algorithms, baseline sentence scoring and default (tie or no polarity words) scoring, and contextual valence shifters. We have implemented a script pipeline that performs opinion classification with these components. It tags and processes an opinion sentence corpus, expands a polarity word lexicon, and then scores


Figure 2: Opinion Classification System Flow
the sentences of the corpus. It does this using the lexicon and contextual valence shifters that modify the polarity contribution of the lexicon words based on various linguistic constructs. Figure 1 lists the data and modules available in our system and Figure 2 describes the flow of inputs and outputs in our script pipeline.

In Figure 2, the opinion sentence corpus is part of speech tagged and then formatted. The formatted corpus is then scored using a polarity word lexicon. There is an optional step of feeding the formatted corpus into one of the polarity lexicon expansion algorithms before performing scoring. These components are detailed in the next sections.

### 3.2 Corpus Processing

Our experiments require a corpus of text annotated at the sentence level for polarity. Our scoring script takes a part-of-speech tagged corpus in XML format where each sentence is given attributes for its annotated score and its feature words. The first phase in our script pipeline is responsible for this processing.

### 3.2.1 Part of Speech Tagging

Part-of-Speech tagging is a common operation in natural language processing [JM00]. It involves assigning a part of speech (e.g. adjective, noun, adverb, verb, preposition) to every word in a corpus, based on a syntactic analysis of the sentence. We experimented with two third party part-of-speech tagging tools, namely NLProcessor [NOO] and the StandfordTagger [S04], which produced similar behavior showing high accuracy. This allowed us to identify lexicon words with the correct part of speech in our corpora.

### 3.3 Baseline Algorithms

Our basic algorithm is based on the one proposed by Hu and Liu [HLO4]. We start with a lexicon of polar adjectives scored as +1 or -1 (eg. amazing has the score +1 , bad has the score -1 ). Sentences are scored by taking the average of the scores of all the lexicon words found in the sentence. The polarity of the opinion expressed in the sentence is the sign of this score. Beyond this basic algorithm, Hu and Liu employ a lexicon expansion step to their 30 hand picked adjectives, adjust word orientation in the presence of negation words (like no, not, and yet), and add heuristics to break ties. The first heuristic applied is to take the average effective opinion of the sentence and relies on the fact that Hu and Liu work on product reviews. This means averaging over the counts of polarity words that are closest to the feature terms in the sentence, rather than all the polarity words. If there is still a tie, the polarity of the previous opinion sentence is taken.

As mentioned above, one aspect that distinguishes Hu and Liu's work from ours is that they are considering names of product features (which they extract before the polarity classification task) as items within the sentence that have a special role (related to the opinion being expressed). This is because the corpus they are using consists of customer reviews of commercial products (like cameras and cell phones), and their task is to recognize opinions toward specific products and their features. Identifying the target of an opinion may prove to be an important factor in opinion analysis, but we have separated this task from that of detecting the polarity of a context already known to express an opinion. We do however include this operation in our set of improvement tools.

The work of Yu and Hatzivassiloglou [YH03] provides a different algorithm. They also start with a polarity lexicon (this time the HM lexicon), and expand this list. For all the nouns, verbs, adjectives, and adverbs in their corpus (which consists of 8000 Wall Street

Journal articles), they compute a polarity score using a log-likelihood equation. Once this expanded lexicon of real-number valued polarities has been generated, it is used to predict the polarity of a sentence by taking the average score for that sentence. We have implemented this algorithm as a component of our infrastructure.

### 3.4 Lexical Polarity and Lexicon Expansion

One of the components of our algorithm is the automatic expansion of an initial seed lexicon to include a greater number of words. This improves the coverage of the algorithm (i.e., more sentences will contain words in the lexicon and will therefore be scored).

### 3.4.1 WordNet

WordNet is a semantic lexicon for English [M+90]. It is a database of dictionary entries along with lists of connections between words. These relationships are of various types, and we are interested in the relations for synonymy and antonymy (sets of words with similar and opposite meaning). Hu Liu describe an algorithm for taking a seed list of polar words and expanding it by adding words found in the corpus that are similar and opposite to the known polar words. The assumption is that similar words carry similar polarity. In WordNet, a given adjective will have a list of senses. Each sense will be associated with a set of synonyms called a synset, and sometimes also two other more loosely associated sets - the see-also set and the similar-to set. For example, the adjective fast has ten senses. Sense eight has quick as a synonym, is similar to hurried, and says to see also firm. We include options in our infrastructure for varying which of these sets to draw synonyms from.

### 3.4.2 Log Likelihood

The log-likelihood equation of [YH03] computes polarity based on a collocation assumption: that a positive word co-occurs with other positive words more frequently that with negative ones, and likewise for a negative word. The score for a word $W_{i}$ with part of speech $P O S_{j}$ ( $j$ can be adjective, noun, adverb, or verb) is

$$
L\left(W_{i}, \operatorname{POS}_{j}\right)=\log \left(\frac{F r e q\left(W_{i}, P O S_{j}, A d j_{p}\right)+\varepsilon}{\operatorname{Freq}\left(W_{\text {all }}, \operatorname{POS}_{j}, \operatorname{Adj}_{p}\right)} / \frac{\operatorname{Freq}\left(W_{i}, P O S_{j}, A d j_{n}\right)+\varepsilon}{\operatorname{Freq}\left(W_{\text {all }}, P O S_{j}, A d j_{n}\right)}\right)
$$

where $\operatorname{Freq}\left(W_{\text {all }}, \mathrm{POS}_{j}\right.$, Adj $\left._{p}\right)$ is the collocational frequency of all words $W_{\text {all }}$ of part of speech $P O S_{j}$ with $A d j_{p}$ and $\varepsilon$ is a smoothing constant ( Yu and Hatzivassilogou use a value of 0.5 ). The sign of the score is the word's polarity. The outer fraction within the $\log$ is a ratio of two values for the word: its relative collocation with positive words and its relative collocation with negative words. When the fraction is greater than one, the log will be positive, corresponding to a stronger association with positive words.
Likewise, when it is between zero and one, the log will be negative, corresponding to stronger association with negative words. This process acts as a lexicon expansion step in that it includes all the corpus words in the set, with varying polarity strengths. We can also cut off inclusion below a threshold to avoid deviations due to noise.

### 3.5 Contextual Valence Shifters

One of our goals is to apply more contextual information in calculating a sentence's polarity. Polanyi and Zaenen [PZ04] provide motivation for this task. They discuss a number of linguistic phenomena, from negation to irony to discourse structure, which could shift the polarity that would have otherwise been predicted by only considering the
lexical level. They provide some preliminary ideas on how to implement some of these, but do not report any experimental results.

We would like to better justify their suggestions, and add to this list, by consulting the linguistics literature. Our experimental results section details how we identify some of these shifters in our development corpus and includes descriptions and examples of each.

### 3.6 Coverage and Accuracy

Since our algorithms are based on the presence of scored polar words, not every sentence we evaluate will have a nonzero average score with a definite positive or negative orientation. Both null sentences that contain no scored polar words and zero-score sentences where the average of the scored words is zero are possible. But for results to be comparable across different configurations, we need a way to increase coverage of our scoring to 100 percent. Therefore we employ a series of default scoring algorithms to classify the sentences that do not have a nonzero score. These are detailed in the experimental results section.

## 4 Experiments and Analysis

In this section we perform a series of experiments that show how we arrive at an improved sentence polarity prediction algorithm.

### 4.1 Lexicons and Corpora

There are two main kinds of data we use in our experiments: polarity lexicons and corpora. Polarity lexicons consist of positive and negative words. We have two main collections: the Hatzivassiloglou and McKeown (HM) lexicon, and the General Inquirer (GI) lexicon. HM contains 1336 ( 657 positive, 679 negative) adjectives. These were compiled manually by [HM97]. The lexicon we are referring to as GI is actually two subsets of a collection of words known as the General Inquirer database developed by Stone et al.[S+66]. The two subsets we have extracted from the database are those with tags Pos and Neg, corresponding to positive and negative words. In its raw form, our extracted GI lexicon contains 4207 ( 1914 positive, 2293 negative) words. They are not explicitly part of speech tagged. They instead come with a variable number of other tags (besides Pos and Neg), some of which imply part of speech membership (such as the Noun tag, or the verb tags $D A V$ (verb descriptive of an action), IAV (verb interpretive of an action) and $S V$ (state verb)). Some but not all also have a "comments" section appended to the entry. However, not all words have tags that imply part of speech. Some might have multiple allowed parts of speech that are only suggested by the comments section, and those tags that imply part of speech membership are not given to all words in that category. Because of these inconsistencies, additional processing must be done if the part of speech tags are required, as they are in our work.

We have a number of corpora that will potentially be useful in our polarity prediction task. The first is the "customer review data" of [HLO4] (the HL corpus). This is a collection of commercial product reviews extracted from amazon.com. They are divided among five products (two digital cameras, a cell phone, a dvd player, and an mp3 player), and there is a total of 3908 sentences. There are 1700 of these that contain opinions about features of the products under review (like "digital zoom" or "size"), and they are annotated with scores representing the polarity and strength of the author's opinion toward each feature reference.

Other corpora that we have collected include the movie review database of Pang et al. [ $\mathrm{P}+02]$ which consists of sentence summaries of movie reviews annotated with a score, derived from www.rottentomatoes.com.

### 4.2 Iterative Failure Analysis

Our plan is to first determine what combination of existing techniques provides the best accuracy in the sentence polarity prediction task. We start with reproducing the Hu and Liu algorithm, using their customer review corpus. From there we add various techniques to improve accuracy. In particular, we vary the polarity lexicon and the word scoring method, which involves counting adjectives or using the log-likelihood equation. We also try expanding the lexicons. We can do this by using WordNet synsets or by applying the log-likelihood collocation equation and extracting words with sufficiently polar scores. From these variations we determine the best combination of algorithms and parameters.

In this work we adopt an empirical methodology of failure analysis. This involves the careful analysis of some of the sentences whose opinion polarity we fail to correctly predict. The failures in a subset of the corpus are categorized into basic classes with the
help of our linguistic references and the largest class motivates the addition of a feature to the system in the next iteration. For example, many sentences fail because of a lack of adjectives and the presence of unaccounted-for polar words of other parts of speech. Examples of these are given below. This motivated the investigation of the General Inquirer lexicon which contains non-adjective polar words.

This approach is similar to the technique known as boosting. Boosting is a general method for improving the accuracy of any given learning algorithm. It is in this sense called a metalearner, or ensemble technique [M00]. The idea is to combine many simple and only weakly accurate classifiers (just better than random) into a single highly accurate classifier. The classifiers are trained sequentially, and on the examples most difficult to classify on previous rounds. Boosting was developed by Freund and Schapire [FS96] and they presented it in the form of the algorithm AdaBoost. It has since been proven successful in many learning tasks such as routing, image retrieval and medical diagnosis [S02] and in natural language processing tasks such as part-of-speech tagging and word-sense disambiguation [E+00]. Our approach differs from boosting in that we seek to discover the classifiers in our data, which may include those that have been previously identified as well as new classifiers. We want to establish what the important classifiers of opinion are first before combining them in a more optimal way.

When we analyze the misclassifications, we separate the lexical from the contextual failures - that is, failure caused by polar words and those caused by language that shifts the contribution of a polar word. Using the failures that resulted from ignoring context information as a guide, we hypothesize a collection of contextual valence shifters, based on linguistic knowledge, that are responsible for shifting the polarity being expressed at the lexical level. We then retry the experiment accounting for these contextual adjustments. After obtaining the configuration that gives the greatest increase in accuracy, we apply the system with and without the contextual valence shifters to another corpus to test the range of applicability across domains and to prevent overfitting.

### 4.3 Baseline Experiments

### 4.3.1 Hu-Liu Algorithm

In the following sections, we present tables of results across product review corpora, details of coverage for the concatenated Hu-Liu corpus (all five products), and lists of example failure sentences. We report our experimental procedure as a sequence of interations of failure analysis from which we measure incremental improvements.

In the first phase of our sequence, we implement the Hu-Liu algorithm. We contacted Ming Hu to ask about acquiring their initial seed set of polar words, but were told that the orgininal 30 were unavailable. Instead they were able to provide an intermediate set of 82 words, which we took as our seed set. With this set we perform the WordNet expansion algorithm but find that this produces results significantly under their reported accuracies [HLO4]. We attribute this to sensitivity on the seed set, given that we did not use the same input to the WordNet expansion algorithm as they did. These results are presented in Table 1. In addition to scoring the five sub-corpora separately and computing the flat average as Hu and Liu have done, we compute the weighted average which gives more weight to the accuracies for sets with more sentences. We also run the algorithm on the corpus formed by concatenating the five sets of product reviews for comparison. This is referred to as the HL corpus. The Always Positive column shows the proportion of sentences in that corpus that are annotated as having positive opinion. This is equivalent to the results obtained running the algorithm with an empty lexicon.

| Table 1: Hu-Liu Implementation |  |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: | :---: |
| Product | Accuracy |  |  |  |  |  |  |  |  |  |
|  | Always Positive | Hu-Liu | Longton HL |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  | $183 / 236$ | 0.775 | 0.927 | $174 / 236$ | 0.737 |  |  |  |  |  |
|  | $129 / 159$ | 0.811 | 0.946 | $126 / 159$ | 0.792 |  |  |  |  |  |
| Nokia cell phone | $186 / 258$ | 0.721 | 0.764 | $196 / 258$ | 0.760 |  |  |  |  |  |
| Creative mp3 player | $419 / 706$ | 0.593 | 0.842 | $468 / 706$ | 0.663 |  |  |  |  |  |
| Apex DVD player | $148 / 341$ | 0.434 | 0.730 | $202 / 341$ | 0.592 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Average |  | $\mathbf{0 . 6 6 7}$ | $\mathbf{0 . 8 4 2}$ |  | $\mathbf{0 . 7 0 9}$ |  |  |  |  |  |
| Weighted Average | $1065 / 1700$ | 0.626 |  | $1166 / 1700$ | 0.686 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| HL Corpus | $1065 / 1700$ | 0.626 |  | $1142 / 1700$ | 0.672 |  |  |  |  |  |

### 4.3.2 Feature Based Effective Opinion

The Hu-Liu corpus consists of sentences that contain an opinion about particular features of the products being reviewed. These features are attached explicitly to the sentences, and one of the components of their algorithm is to use these feature words in breaking ties. When a sentence score is found to be zero, the effective opinion of the sentence is computed. This is the average of the polarities of the lexicon words that are closest to the feature words in the sentence. Furthermore, there is an additional set of logic that is used in the presence of but-conjunctions. The effective opinion of the but-clause takes priority over the overall sentence score. We separate out these two levels of effective opinion usage as separate options in our implementation.

To see how the various pieces of the Hu-Liu algorithm contribute to the overall accuracy, Table 2 shows the accuracies achieved when sequentially enabling different parts of the algorithm. Each configuration is specified by an abbreviated term of the lexicon used followed by the set of options given to the scoring script. The Hu-Liu lexicon expanded with WordNet is abbreviated hulexWN. Option $j$ refers to adjectives being the part of speech that are candidates for scoring (later we will see $n, r$, and $v$ stand of noun, adverb,
and verb). Option $g$ means using the negation adjustment and $d$ means using a list of stop words across which negation is inactive. This helps increase accuracy by effectively establishing syntactic boundaries. Options $e$ and $b$ refer to the effective opinion tie breaking and effective opinion but-clause components of the Hu -Liu algorithm. The configuration hulexWN -jdgeb corresponds to our implementation of the reported Hu -Liu experiment (called Longton HL above). As seen in Table 2, negation usually improves accuracy, while only some products benefit from the effective opinion options.

| Product | Accuracy |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hulexWN-j |  | hulexWN -jdg |  | hulexWN -jdge |  | hulexWN -jdgeb |  |
|  |  |  |  |  |  |  |  |  |
| Apex player | 184/341 | 0.5396 | 186/341 | 0.5455 | 180/341 | 0.5279 | 202/341 | 0.5924 |
| Canon camera | 180/236 | 0.7627 | 179/236 | 0.7585 | 178/236 | 0.7542 | 174/236 | 0.7373 |
| Creative mp3 | 457/706 | 0.6473 | 486/706 | 0.6884 | 478/706 | 0.6771 | 468/706 | 0.6629 |
| Nikon camera | 131/159 | 0.8239 | 134/159 | 0.8428 | 135/159 | 0.8491 | 126/159 | 0.7925 |
| Nokia cell | 194/258 | 0.7519 | 199/258 | 0.7713 | 191/258 | 0.7403 | 196/258 | 0.7597 |
| Average |  | 0.7051 |  | 0.7213 |  | 0.7097 |  | 0.7089 |
| W. Average | 1146/1700 | 0.6741 | 1184/1700 | 0.6965 | 1162/1700 | 0.6835 | 1166/1700 | 0.6859 |
| HL Corpus | 1135/1700 | 0.6677 | 1169/1700 | 0.6877 | 1151/1700 | 0.6771 | 1142/1700 | 0.6718 |

Table 3: HulexWN - Coverage for HL Corpus

| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hulexWN-j |  | hulexWN -jdg |  | hulexWN -jdge |  | hulexWN -jdgeb |  |
|  |  |  |  |  |  |  |  |  |
| Total | 1135/1700 | 0.6677 | 1169/1700 | 0.6877 | 1151/1700 | 0.6771 | 1142/1700 | 0.6718 |
| Nonnull | 758/1037 | 0.7310 | 782/1037 | 0.7541 | 779/1037 | 0.7512 | 761/1037 | 0.7339 |
| Nonzero | 716/957 | 0.7482 | 728/950 | 0.7663 | 764/1019 | 0.7498 | 739/1009 | 0.7324 |
| Zero | 42/80 | 0.5250 | 54/87 | 0.6207 | 15/18 | 0.8333 | 22/28 | 0.7857 |
| Null | 377/663 | 0.5686 | 387/663 | 0.5837 | 372/663 | 0.5611 | 381/663 | 0.5747 |
| NonzeroReal |  |  |  |  | 728/950 | 0.7663 |  |  |
| NonzeroEff |  |  |  |  | 36/69 | 0.5217 |  |  |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 1037/1700 | 0.6100 |  |  |  |  |  |  |
| Nonzero | 957/1700 | 0.5629 | 950/1700 | 0.5588 | 1019/1700 | 0.5994 | 1009/1700 | 0.5935 |

Table 3 provides a breakdown of the scoring based on lexicon coverage. This breakdown is given for the concatenated HL Corpus as a summary and this pattern is continued in
subsequent sections. In the Accuracies section, Total is the sum of three mutually exclusive subsets - Nonzero, Zero, and Null. Null sentences are those that do not contain any lexicon words. Nonzero and Zero are sentences that contain lexicon words and have a nonzero or zero score, respectively (their union forms Nonnull). When the $e$ option is used, tie breaking is attempted on Zero sentences. If a nonzero effective opinion score is found, the sentence counts as Nonzero. In this case Nonzero is further divided into those that were nonzero initially and those that were due to effective opinion tie breaking. This breakdown is reported by the NonzeroReal and NonzeroEff sets. There are two coverage measurements listed. Nonnull is the fraction of sentences containing lexicon words and Nonzero is the fraction of sentences with a nonzero score. Nonnull coverages that are the same as the configuration to their left are omitted to highlight that the configurations contain the same lexicon words.

### 4.3.3 Polysemy and Bipolarity in WordNet Expansion

Investigating the failures, we find that many are due to wrongly scored WordNet generated adjectives. Often these are words that are either polysemous as both polar and neutral words or even as both positive and negative words. Here are some examples with the scored words marked.

- The screen is large(-), defined, and easy to read, and the silver unit is naturally cool.
- The locations of various(+) buttons on one side or the other is somewhat illogical.

In the first sentence, large is being used to express a positive opinion, but was added to our lexicon via its synonym big, which was added as a synonym of our negative seed word bad. The third sense of bad listed in WordNet is a synonym of big, as in "a bad storm". The second sentence is negative, but the neutral word various was incorrectly given a positive score via its synonym versatile.

We try variations on the parameters of the WordNet algorithm. These include the different possible sets to draw synonyms from (the see-also, synset, and similar-to sets), and the number of words to use from the set starting at the beginning (since earlier listed ones often are the more common and sometimes less ironic synonyms).

These didn't improve results significantly so we consulted Ming Hu on their choices. They confirmed that they used only the synset with no restriction on the number of senses. Our results with these parameters don't improve our accuracy.

### 4.4 Lexical Polarity Analysis

WordNet expansion introduces a significant amount of noise to our lexicon. To see if the small seed set provided by Hu and Liu was simply too sparse or if WordNet expansion helps at all for a larger seed set, we investigate the use of a larger manually constructed lexicon of polarity words.

### 4.4.1 Intelligent HM Lexicon

We run our scoring algorithm on the Hu-Liu corpus this time using Hatzivassiloglou and McKeown's manually constructed lexicon of 1336 positive and negative words. This intelligent lexicon outperforms Hu-Liu's WordNet expanded seed set, as seen in Tables 5 and 6. We even find that WordNet expansion negatively affects HM (adding mostly wrong-scored words). This is reported in Table 4. We find that with the intelligent lexicon, effective opinion tie breaking does not improve results over the default previous sentence score. To measure statistical significance of the improvement in accuracies across our subcorpora, we perform a two-tailed paired t-test (described in [C95]) on our
best HM result and the Hu -Liu baseline with a significance level of 0.05 . We obtain a pvalue of 0.032 which shows that the increased accuracy is significant.

Table 4: HM Lexicon with WordNet Expansion

| Product | Accuracy |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hmWN-j |  | hmWN -jdg |  | hmWN -jdge |  | hmWN -jdgeb |  |
|  |  |  |  |  |  |  |  |  |
| Apex player | 199/341 | 0.5836 | 212/341 | 0.6217 | 219/341 | 0.6422 | 219/341 | 0.6422 |
| Canon camera | 180/236 | 0.7627 | 182/236 | 0.7712 | 180/236 | 0.7627 | 175/236 | 0.7415 |
| Creative mp3 | 461/706 | 0.6530 | 493/706 | 0.6983 | 494/706 | 0.6997 | 474/706 | 0.6714 |
| Nikon camera | 124/159 | 0.7799 | 125/159 | 0.7862 | 120/159 | 0.7547 | 110/159 | 0.6918 |
| Nokia cell | 195/258 | 0.7558 | 197/258 | 0.7636 | 195/258 | 0.7558 | 191/258 | 0.7403 |
|  |  |  |  |  |  |  |  |  |
| Average |  | 0.7070 |  | 0.7282 |  | 0.7230 |  | 0.6975 |
| W. Average | 1159/1700 | 0.6818 | 1209/1700 | 0.7112 | 1208/1700 | 0.7106 | 1169/1700 | 0.6876 |
| HL Corpus | 1144/1700 | 0.6729 | 1189/1700 | 0.6994 | 1191/1700 | 0.7006 | 1154/1700 | 0.6788 |

Table 5: HM Lexicon without WordNet

| Product | Accuracy |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hm-j |  | hm -jdg |  | hm -jdge |  | hm-jdgeb |  |
|  |  |  |  |  |  |  |  |  |
| Apex player | 206/341 | 0.6041 | 216/341 | 0.6334 | 218/341 | 0.6393 | 218/341 | 0.6393 |
| Canon camera | 189/236 | 0.8009 | 188/236 | 0.7966 | 188/236 | 0.7966 | 183/236 | 0.7754 |
| Creative mp3 | 475/706 | 0.6728 | 511/706 | 0.7238 | 509/706 | 0.7210 | 493/706 | 0.6983 |
| Nikon camera | 128/159 | 0.8050 | 128/159 | 0.8050 | 122/159 | 0.7673 | 117/159 | 0.7359 |
| Nokia cell | 198/258 | 0.7674 | 198/258 | 0.7674 | 196/258 | 0.7597 | 190/258 | 0.7364 |
|  |  |  |  |  |  |  |  |  |
| Average |  | 0.7300 |  | 0.7453 |  | 0.7368 |  | 0.7171 |
| W. Average | 1196/1700 | 0.7035 | 1241/1700 | 0.7300 | 1233/1700 | 0.7253 | 1201/1700 | 0.7065 |
| HL Corpus | 1196/1700 | 0.7035 | 1241/1700 | 0.7300 | 1233/1700 | 0.7253 | 1201/1700 | 0.7065 |

Table 6: HM - Coverage for HL Corpus

| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hm -j |  | hm -jdg |  | hm - jdge |  | hm -jdgeb |  |
|  |  |  |  |  |  |  |  |  |
| Total | 1196/1700 | 0.7035 | 1241/1700 | 0.7300 | 1233/1700 | 0.7253 | 1201/1700 | 0.7065 |
| Nonnull | 736/923 | 0.7974 | 760/923 | 0.8234 | 758/923 | 0.8212 | 720/923 | 0.7801 |
| Nonzero | 696/852 | 0.8169 | 722/849 | 0.8504 | 750/906 | 0.8278 | 706/896 | 0.7880 |
| Zero | 40/71 | 0.5634 | 38/74 | 0.5135 | 8/17 | 0.4706 | 14/27 | 0.5185 |
| Null | 460/777 | 0.5920 | 481/777 | 0.6191 | 475/777 | 0.6113 | 481/777 | 0.6191 |
| NonzeroReal |  |  |  |  | 722/849 | 0.8504 |  |  |
| NonzeroEff |  |  |  |  | $28 / 57$ | 0.4912 |  |  |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 923/1700 | 0.5429 |  |  |  |  |  |  |
| Nonzero | 852/1700 | 0.5012 | 849/1700 | 0.4994 | 906/1700 | 0.5329 | 896/1700 | 0.5271 |

### 4.4.2 Polarity in other Parts of Speech and the GI Lexicon

Failure analysis of the results using HM reveals the next major source of error: failure due to sparsity of adjectives and the non-scoring of clearly polar non-adjective words. The following are some examples:

- It would only transfer 30 or so songs, and then come up with an error(-).
- I gave it only 3 stars due to the fact that the 1st one broke(-) when I dropped it from a fairly short distance (less than 2 ft).
- The "scene" mode works well(+) for the remainder of shots that are not going to be in a "regular" setting.

These sentences would be scored correctly if the marked noun, verb, and adverb were included in our lexicon. To address this, we test the General Inquirer (GI) lexicon. We develop the best possible tagged form we can manage automatically, which we call the "XGI" (extended GI). Each word is given all of the adjective, noun, adverb, and verb parts of speech, at the risk of some error due to polarity-disagreeing polysemy across parts of speech for a given word. The results show an increase in coverage as well as the potential of all of the added parts of speech helping to improve accuracy. The adjectives from XGI are tested alone for comparison to HM in Table 7, and Table 8 shows our tests of the different parts of speech in isolation and in combination. All four produces the best accuracies so far (the best being 0.786) and this is reported in Tables 9 and 10.

| HL Corpus | Accuracy |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | xgi - j |  | xgi -jdg |  | xgi -jdge |  |
|  | 1185/1700 | 0.6971 | 1230/1700 | 0.7235 | 1218/1700 | 0.7165 |

Table 8: XGI Lexicon - Other Parts of Speech

| HL Corpus | Accuracy |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | xgi-j |  | xgi-n |  | xgi -r |  | xgi-v |  |
|  | 1185/1700 | 0.6971 | 989/1700 | 0.5818 | 1039/1700 | 0.6112 | 880/1700 | 0.5177 |
|  | xgi-jdg |  | xgi -ndg |  | xgi -rdg |  | xgi -vdg |  |
|  | 1230/1700 | 0.7235 | 1006/1700 | 0.5918 | 1047/1700 | 0.6159 | 985/1700 | 0.5794 |
|  | xgi-jn |  | xgi -jr |  | xgi -jnr |  | xgi -nrv |  |
|  | 1183/1700 | 0.6959 | 1191/1700 | 0.7006 | 1208/1700 | 0.7106 | 1071/1700 | 0.6300 |

Table 9: XGI Lexicon - All Parts of Speech

| Product | Accuracy |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | xgi-jnrv |  | xgi -jnrvdg |  | xgi-jnrvdge |  |
|  |  |  |  |  |  |  |
| Apex player | 211/341 | 0.6188 | 234/341 | 0.6862 | 242/341 | 0.7097 |
| Canon camera | 198/236 | 0.8390 | 198/236 | 0.8390 | 194/236 | 0.8220 |
| Creative mp3 | 468/706 | 0.6629 | 512/706 | 0.7252 | 509/706 | 0.7210 |
| Nikon camera | 134/159 | 0.8428 | 138/159 | 0.8679 | 139/159 | 0.8742 |
| Nokia cell | 202/258 | 0.7830 | 208/258 | 0.8062 | 207/258 | 0.8023 |
|  |  |  |  |  |  |  |
| Average |  | 0.7493 |  | 0.7849 |  | 0.7858 |
| W. Average | 1213/1700 | 0.7135 | 1290/1700 | 0.7588 | 1291/1700 | 0.7594 |
| HL Corpus | 1213/1700 | 0.7135 | 1290/1700 | 0.7588 | 1291/1700 | 0.7594 |


| Accuracies | Configuration |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | xgi-jnrv |  | xgi -jnrvdg |  | xgi -jnrvdge |  |
|  |  |  |  |  |  |  |
| Total | 1213/1700 | 0.7135 | 1290/1700 | 0.7588 | 1291/1700 | 0.7594 |
| Nonnull | 995/1337 | 0.7442 | 1059/1337 | 0.7921 | 1059/1337 | 0.7921 |
| Nonzero | 920/1201 | 0.7660 | 973/1193 | 0.8156 | 1045/1313 | 0.7959 |
| Zero | 75/136 | 0.5515 | 86/144 | 0.5972 | 14/24 | 0.5833 |
| Null | 218/363 | 0.6006 | 231/363 | 0.6364 | 232/363 | 0.6391 |
| NonzeroReal |  |  |  |  | 973/1193 | 0.8156 |
| NonzeroEff |  |  |  |  | 72/120 | 0.6000 |
| Coverage |  |  |  |  |  |  |
| Nonnull | 1337/1700 | 0.7865 |  |  |  |  |
| Nonzero | 1201/1700 | 0.7065 | 1193/1700 | 0.7018 | 1313/1700 | 0.7724 |

### 4.4.3 Combined Lexicon

We also test the union of the GI lexicon with the HM lexicon and find the best results yet (0.793). We note however a few inconsistencies between the two lexicons. GI had both
a positive and negative entry for help: $V$, and HM and GI had a conflict in polarity for flashy: $J$ (positive in HM and negative in GI). We removed these words in constructing the union. Tables 11 and 12 report the results. Performing the two-tailed paired $t$-test this time between the HM and Combined lexicon results yields a p-value of 0.011 , again showing a statistically significant increase in accuracy.

| Product | Accuracy |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hx -jnrv |  | hx -jnrvdg |  | hx -jnrydge |  |
|  |  |  |  |  |  |  |
| Apex player | 219/341 | 0.6422 | 246/341 | 0.7214 | 249/341 | 0.7302 |
| Canon camera | 201/236 | 0.8517 | 197/236 | 0.8348 | 194/236 | 0.8220 |
| Creative mp3 | 482/706 | 0.6827 | 530/706 | 0.7507 | 526/706 | 0.7450 |
| Nikon camera | 134/159 | 0.8428 | 134/159 | 0.8428 | 134/159 | 0.8428 |
| Nokia cell | 205/258 | 0.7946 | 210/258 | 0.8140 | 211/258 | 0.8178 |
|  |  |  |  |  |  |  |
| Average |  | 0.7628 |  | 0.7927 |  | 0.7916 |
| W. Average | 1241/1700 | 0.7300 | 1317/1700 | 0.7747 | 1314/1700 | 0.7729 |
| HL Corpus | 1241/1700 | 0.7300 | 1317/1700 | 0.7747 | 1314/1700 | 0.7729 |


| Accuracies | Configuration |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hx -jnrv |  | hx -jnrvdg |  | hx -jnrvdge |  |
|  |  |  |  |  |  |  |
| Total | 1241/1700 | 0.7300 | 1317/1700 | 0.7747 | 1314/1700 | 0.7729 |
| Nonnull | 1065/1416 | 0.7521 | 1135/1416 | 0.8016 | 1134/1416 | 0.8009 |
| Nonzero | 963/1242 | 0.7754 | 1028/1247 | 0.8244 | 1116/1386 | 0.8052 |
| Zero | 102/174 | 0.5862 | 107/169 | 0.6331 | 18/30 | 0.6000 |
| Null | 176/284 | 0.6197 | 182/284 | 0.6409 | 180/284 | 0.6338 |
| NonzeroReal |  |  |  |  | 1028/1247 | 0.8244 |
| NonzeroEff |  |  |  |  | 88/139 | 0.6331 |
| Coverage |  |  |  |  |  |  |
| Nonnull | 1416/1700 | 0.8329 |  |  |  |  |
| Nonzero | 1242/1700 | 0.7306 | 1247/1700 | 0.7335 | 1386/1700 | 0.8153 |

### 4.4.4 Nonpolar and Bipolar Words and the Robust Lexicon

The next iteration of failure analysis reveals nonpolar and bipolar words in this combined lexicon. Polarity of words across their different senses varies and so given the context, some of our lexicon words are either used in a way that does not carry the labeled polarity or indeed carries the opposite polarity. Examples of these are as follows:

- The main(+) problem with the Nomad Jukebox Zen Xtra 30GB is the software.
- It looks very cool(-), and seems quite small(-) to me and very light.

In the first sentence main is neutral instead of positive, and in the second sentence both cool and small are positive instead of negative. We subjectively remove words from the subset found in the corpus that are deemed nonpolar or bipolar, but without direct reference to the failures so as to partially but not exhaustively prune the combined lexicon in a subjective but domain-independent fashion. This robust lexicon significantly improves performance to 0.835 . The complete results are shown in Tables 13 and 14 .

| Table 13: Robust Lexicon myHX |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Product | Accuracy |  |  |  |  |  |  |  | myhx -jnrvdg |  |  |  |  | myhx -jnrvdge |  |
|  | myhx -jnrv |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 0.7302 |  |  |  |  |  |  |  |  |  |  |
|  | $228 / 341$ | 0.6686 | $245 / 341$ | 0.7185 | $249 / 341$ | 0.73025 |  |  |  |  |  |  |  |  |  |
| Canon camera | $200 / 236$ | 0.8475 | $209 / 236$ | 0.8856 | $213 / 236$ | 0.9025 |  |  |  |  |  |  |  |  |  |
| Creative mp3 | $510 / 706$ | 0.7224 | $570 / 706$ | 0.8074 | $565 / 706$ | 0.8003 |  |  |  |  |  |  |  |  |  |
| Nikon camera | $135 / 159$ | 0.8491 | $140 / 159$ | 0.8805 | $137 / 159$ | 0.8616 |  |  |  |  |  |  |  |  |  |
| Nokia cell | $219 / 258$ | 0.8488 | $228 / 258$ | 0.8837 | $225 / 258$ | 0.8721 |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Average |  | 0.7873 |  | 0.8351 |  | 0.8334 |  |  |  |  |  |  |  |  |  |
| W. Average | $1292 / 1700$ | 0.7600 | $1392 / 1700$ | 0.8188 | $1389 / 1700$ | 0.8171 |  |  |  |  |  |  |  |  |  |
| HL Corpus | $1291 / 1700$ | 0.7594 | $1392 / 1700$ | 0.8188 | $1389 / 1700$ | 0.8171 |  |  |  |  |  |  |  |  |  |


| Table 14: MyHX - Coverage for HL Corpus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Accuracies | Configuration |  |  |  |  |  |  | myhx -jnrvdg |  |  |  |  | myhx -jnrvdge |  |
|  | myhx -jnrv |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $1291 / 1700$ | 0.7594 | $1392 / 1700$ | 0.8188 | $1389 / 1700$ | 0.8171 |  |  |  |  |  |  |  |  |
|  | $961 / 1154$ | 0.8328 | $1028 / 1154$ | 0.8908 | $1029 / 1154$ | 0.8917 |  |  |  |  |  |  |  |  |
| Nonzero | $912 / 1069$ | 0.8531 | $978 / 1072$ | 0.9123 | $1021 / 1143$ | 0.8933 |  |  |  |  |  |  |  |  |
| Zero | $49 / 85$ | 0.5765 | $50 / 82$ | 0.6098 | $8 / 11$ | 0.7273 |  |  |  |  |  |  |  |  |
| Null | $330 / 546$ | 0.6044 | $364 / 546$ | 0.6667 | $360 / 546$ | 0.6593 |  |  |  |  |  |  |  |  |
| NonzeroReal |  |  |  |  | $978 / 1072$ | 0.9123 |  |  |  |  |  |  |  |  |
| NonzeroEff |  |  |  |  | $43 / 71$ | 0.6056 |  |  |  |  |  |  |  |  |
| Coverage |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Nonnull | $1154 / 1700$ | 0.6788 |  |  |  |  |  |  |  |  |  |  |  |  |
| Nonzero | $1069 / 1700$ | 0.6288 | $1072 / 1700$ | 0.6306 | $1143 / 1700$ | 0.6724 |  |  |  |  |  |  |  |  |

### 4.4.5 The Augmented Lexicon

Failure analysis then reveals a number of remaining errors due to missing scored words. The following examples show words that are missing from the robust lexicon that are needed to predict the sentence's opinion. They are marked with the needed polarity.

- The included lens cap is very loose(-) on the camera.
- The cool(+) thing about the ad-2600 is that it plays a lot of different file types.

We finally top-up the lexicon with these domain-dependent words in order to reduce our error set to those errors of the more rare but more sophisticated contextual type.

Accuracies improve accordingly and are reported in Tables 15 and 16. We perform the ttest between the Combined and Augmented lexicon and obtain a p-value of 0.027 , showing another statistically significant increase in accuracy.

| Table 15: Augmented Lexicon myHX+ |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: |
| Product | Accuracy |  |  |  |  |  |  |  |  |
|  | myhx+-jnrv |  | myhx+-jnrvdg |  |  |  |  | myhx+-jnrvdge |  |
|  |  |  |  |  |  |  |  |  |  |
|  | $235 / 341$ | 0.6892 | $255 / 341$ | 0.7478 | $259 / 341$ | 0.7595 |  |  |  |
| Canon camera | $207 / 236$ | 0.8771 | $214 / 236$ | 0.9068 | $218 / 236$ | 0.9237 |  |  |  |
| Creative mp | $521 / 706$ | 0.7380 | $575 / 706$ | 0.8145 | $571 / 706$ | 0.8088 |  |  |  |
| Nikon camera | $136 / 159$ | 0.8554 | $141 / 159$ | 0.8868 | $138 / 159$ | 0.8679 |  |  |  |
| Nokia cell | $218 / 258$ | 0.8450 | $227 / 258$ | 0.8798 | $226 / 258$ | 0.8760 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Average |  | 0.8009 |  | 0.8471 |  | 0.8472 |  |  |  |
| W. Average | $1317 / 1700$ | 0.7747 | $1412 / 1700$ | 0.8306 | $1412 / 1700$ | 0.8306 |  |  |  |
| HL Corpus | $1316 / 1700$ | 0.7741 | $1412 / 1700$ | 0.8306 | $1412 / 1700$ | 0.8306 |  |  |  |


| Table 16: MyHX - Coverage for HL Corpus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Accuracies | Configuration |  |  |  |  |  |  | myhx+-jnrvdg |  |  |  |  | myhx+-jnrvdge |  |
|  | myhx+-jnrv |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $1316 / 1700$ | 0.7741 | $1412 / 1700$ | 0.8306 | $1412 / 1700$ | 0.8306 |  |  |  |  |  |  |  |  |
|  | $1019 / 1211$ | 0.8415 | $1088 / 1211$ | 0.8984 | $1092 / 1211$ | 0.9017 |  |  |  |  |  |  |  |  |
| Nonzero | $971 / 1128$ | 0.8608 | $1039 / 1130$ | 0.9195 | $1084 / 1201$ | 0.9026 |  |  |  |  |  |  |  |  |
| Zero | $48 / 83$ | 0.5783 | $49 / 81$ | 0.6049 | $8 / 10$ | 0.8000 |  |  |  |  |  |  |  |  |
| Null | $297 / 489$ | 0.6074 | $324 / 489$ | 0.6626 | $320 / 489$ | 0.6544 |  |  |  |  |  |  |  |  |
| NonzeroReal |  |  |  |  | $1039 / 1130$ | 0.9195 |  |  |  |  |  |  |  |  |
| NonzeroEff |  |  |  |  | $45 / 71$ | 0.6338 |  |  |  |  |  |  |  |  |
| Coverage |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Nonnull | $1211 / 1700$ | 0.7865 |  |  |  |  |  |  |  |  |  |  |  |  |
| Nonzero | $1128 / 1700$ | 0.7865 | $1130 / 1700$ | 0.7865 | $1201 / 1700$ | 0.7865 |  |  |  |  |  |  |  |  |

### 4.5 Contextual Valence Shifters

The next stage involves the study of the effectiveness of the valence shifting operations performed by the Hu -Liu algorithm [HL04], an implementation of some of the shifters suggested by Polanyi and Zaenen [PZ04], and some newly classified valence shifters. Hu -Liu applies polarity shifting logic for negation and handles but-conjunction in a special way. Polanyi and Zaenen suggest the shifting capacity of modals and presupposition. Finally, we discover cases where contrastives, adverbs of excess and sufficiency, and hedging can shift polarity. We also note that the conjunction but is
common in reviews but we discover that its usage is complex and varied, and its other usages are more frequent than its negating behavior.

We experiment with various parameters to the shifters. One is the subset of the sentence that is affected, including attention to stop-words that form syntactic boundaries. Another is the operational word sets that are associated with each shifter context. Accuracies improve sequentially as we tune each shifter and apply them to our corpus with our various lexicons.

We begin with refining the negation shifter. By varying the keyword set and the window of activity, we tune these parameters to optimal values. Including the word no helps now with nouns present and a window of six words instead of Hu and Liu's five optimizes the results. These improvements are summarized in Table 17.

| Accuracies | Configuration |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | myhx+ -jnry |  | myhx+-jnrvdg |  | myhx+-jnrvdge |  |
|  |  |  |  |  |  |  |
| Total | 1316/1700 | 0.7741 | 1425/1700 | 0.8382 | 1426/1700 | 0.8388 |
| Coverage |  |  |  |  |  |  |
| Nonnull | 1211/1700 | 0.7124 |  |  |  |  |
| Nonzero | 1128/1700 | 0.6635 | 1128/1700 | 0.6635 | 1201/1700 | 0.70647 |

Analyzing the failures from the tuned augmented lexicon experiment, we note the presence of various polarity shifting constructs. One of these is where a comparison is made between the main subject of a sentence and another subject for the sake of contrast. Polar words used in the auxiliary clause are usually of opposite polarity to the opinion being expressed toward the main subject. These clauses are often marked with a contrastive keyword like although or despite, so we develop a shifter pattern called a contrastive to adjust sentence polarity in the presence of such clauses. The following examples benefit from modeling the contrastive shifter by inverting the polarity of the lexicon words within the contrastive clause.

- Despite(c) this minor disappointment(-), I highly recommend the Canon G3 to anyone who is serious about digital photography.
- Although(c) I find it more convenient(+) to use 1-touch dialing, this phone does not have voice dialing.

Another polarity shifting construct is found in the use of modality. This is where an evaluation of a subject is suggested if some other condition had been true, thus conveying the opposite opinion in that condition's absence. These constructs are marked with modal words like would and should, and most commonly shift opinion when followed by be or have been. Predicting the polarity of the following sentences fails without modeling the inverting function of modal phrases.

- A spare battery would have been $(m)$ great (+).
- At the very least, a sturdier more protective carrying case would be(m) nice(+).

The lexicon words are labeled with their polarity, which needs to be inverted by the modifying modal phrase.

Sentence polarity can also be shifted by the use of presupposition. This occurs when a polar word is modified by a word that presupposes an expectation that was not met. The following are examples that require an inversion of polarity based on the presence of presuppositional terms.

- It's way less(p) expensive(-) than the iPod.
- The front panel refused(p) to clip in correctly(+), leaving a noticeable gap between the panel and base of the player.
- Forget( $p$ ) about the sleek(+) looks if it can't play some of your real dvds.
- The remote is a little hard to(p) understand( + ).

We developed shifter functionality into our algorithm for each of these constructs and the results are summarized by Tables 18 and 19. We see an increase in accuracy for each
construct individually and when used in conjunction with one another (accuracy is 0.868 ). Also we note that although effective opinion tie breaking improves coverage it does not improve accuracy in these cases. Performing the two-tailed paired t-test now on the results of the tuned Augmented lexicon with and without contextual valence shifters yields a p-value of 0.028 . This shows another statistically significant increase in accuracy.

| Table 18: Augmented Lexicon MyHX+ with Shifters |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
| Product | Accuracy |  |  |  |
|  | myhx+-jnrvdgcmp | myhx+-jnrvdgcmpe |  |  |
|  |  |  |  |  |
|  | $266 / 341$ | 0.7801 | $268 / 341$ | 0.7859 |
| Canon camera | $215 / 236$ | 0.9110 | $218 / 236$ | 0.9237 |
| Creative mp3 | $593 / 706$ | 0.8399 | $589 / 706$ | 0.8343 |
| Nikon camera | $142 / 159$ | 0.8931 | $140 / 159$ | 0.8805 |
| Nokia cell | $236 / 258$ | 0.9147 | $231 / 258$ | 0.8954 |
|  |  |  |  |  |
| Average |  | $\mathbf{0 . 8 6 7 8}$ |  | 0.8642 |
| W. Average | $1452 / 1700$ | 0.8541 | $1446 / 1700$ | 0.8506 |
| HL Corpus. | $\mathbf{1 4 5 2 / 1 7 0 0}$ | 0.8541 | $1446 / 1700$ | 0.8506 |

Table 19: MyHX+ with Shifters - Coverage for HL Corpus

| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | myhx+-jnrvdgc |  | myhx+-jnrvdgce |  | myhx+-jnrvdgm |  | myhx+-jnrvdgme |  |
|  |  |  |  |  |  |  |  |  |
| Total | 1431/1700 | 0.8418 | 1430/1700 | 0.8412 | 1431/1700 | 0.8418 | 1430/1700 | 0.8412 |
| Nonnull | 1099/1211 | 0.9075 | 1104/1211 | 0.9116 | 1102/1211 | 0.9100 | 1107/1211 | 0.9141 |
| Nonzero | 1047/1128 | 0.9282 | 1096/1200 | 0.9133 | 1049/1127 | 0.9308 | 1097/1199 | 0.9149 |
| Zero | 52/83 | 0.6265 | 8/11 | 0.7273 | 53/84 | 0.6310 | 10/12 | 0.8333 |
| Null | 332/489 | 0.6789 | 326/489 | 0.6667 | 329/489 | 0.6728 | 323/489 | 0.6605 |
| NonzeroReal |  |  | 1047/1128 | 0.9282 |  |  | 1049/1127 | 0.9308 |
| NonzeroEff |  |  | 49/72 | 0.6806 |  |  | 48/72 | 0.6667 |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 1211/1700 | 0.7124 |  |  |  |  |  |  |
| Nonzero | 1128/1700 | 0.6635 | 1200/1700 | 0.7059 | 1127/1700 | 0.6629 | 1199/1700 | 0.7053 |
|  |  |  |  |  |  |  |  |  |
|  | Configurat |  |  |  |  |  |  |  |
|  | myhx+-jnr | dgp | myhx+-jnr | dgpe | myhx+-jnrv | dgcmp | myhx+-jnrv | dgcmpe |
| Accuracies |  |  |  |  |  |  |  |  |
| Total | 1440/1700 | 0.8471 | 1438/1700 | 0.8459 | 1452/1700 | 0.8541 | 1446/1700 | 0.8506 |
| Nonnull | 1107/1211 | 0.9141 | 1112/1211 | 0.9183 | 1114/1211 | 0.9199 | 1115/1211 | 0.9207 |
| Nonzero | 1059/1133 | 0.9347 | 1105/1203 | 0.9185 | 1061/1132 | 0.9373 | 1106/1200 | 0.9217 |
| Zero | 48/78 | 0.6154 | $7 / 8$ | 0.8750 | 53/79 | 0.6709 | 9/11 | 0.8182 |
| Null | 333/489 | 0.6810 | 326/489 | 0.6667 | 338/489 | 0.6912 | 331/489 | 0.6769 |
| NonzeroReal |  |  | 1059/1133 | 0.9347 |  |  | 1061/1132 | 0.9373 |
| NonzeroEff |  |  | 46/70 | 0.6571 |  |  | 45/68 | 0.6618 |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 1211/1700 | 0.7124 |  |  |  |  |  |  |
| Nonzero | 1133/1700 | 0.6665 | 1203/1700 | 0.7077 | 1132/1700 | 0.6659 | 1200/1700 | 0.7059 |

We then go back to our combined and robust lexicons and run these now with our full array of shifters. Results are shown in Table 20.

Table 20: HX and myHX with Shifters - Coverage for HL Corpus

| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hx -jnrvdg |  | hx -jnrvdgcmp |  | myhx -jnrvdg |  | myhx -jnrvdgcmp |  |
|  |  |  |  |  |  |  |  |  |
| Total | 1331/1700 | 0.7829 | 1336/1700 | 0.7859 | 1408/1700 | 0.8282 | 1432/1700 | 0.8424 |
| Nonnull | 1148/1416 | 0.8107 | 1158/1413 | 0.8195 | 1037/1154 | 0.8986 | 1053/1154 | 0.9125 |
| Nonzero | 1036/1241 | 0.8348 | 1045/1240 | 0.8427 | 987/1071 | 0.9216 | 999/1072 | 0.9319 |
| Zero | 112/175 | 0.6400 | 113/173 | 0.6532 | 50/83 | 0.6024 | 54/82 | 0.6585 |
| Null | 183/284 | 0.6444 | 178/287 | 0.6202 | 371/546 | 0.6795 | $379 / 546$ | 0.6941 |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 1416/1700 | 0.8329 | 1413/1700 | 0.8312 | 1154/1700 | 0.6788 | 1154/1700 | 0.6788 |
| Nonzero | 1241/1700 | 0.7300 | 1240/1700 | 0.7294 | 1071/1700 | 0.6300 | 1072/1700 | 0.6306 |

We note that accuracies are increasingly improved by the shifters as the lexicon becomes more refined.

There are a number of other kinds of contextual valence shifters that we have discovered and experimented with in the Hu Liu corpus. One is where the conjunction but, or synonyms except and however, are used to contrast two subjects. When they are used for this function, they are similar to the class of contrastive shifters we identified but follow rather than precede the polar word they are shifting. We found that although these shifters are responsible for a number of failures, the negating function of these conjunctions is only one of their uses. They are even more often used in ways that do not directly shift the polarity of the sentence away from the preceding polar words. Thus they are not by themselves effective shifters in the Hu Liu corpus, producing more false positives than successful polarity shifts. These sentences are examples of butconjunctions acting as negating shifters, where the annotated sentence polarity is given at the beginning of the line:

- (-) Nice(+) machines, but(b) I consider their quality pretty low now.
- (+)I was hesitant (-) given the price, but(b) I've been extremely impressed since receiving it.

These sentences are examples where but is not shifting the sentence's polarity and would result in a failed prediction:

- (+) Basic usage is easy(+), but(b) the remote has a lot of buttons that I haven't used.
- (-) A little weighty(-) (9 ounces... no biggie), but(b) otherwise fine.

Another shifter construct responsible for failures is the use of a class of adverbs known as adverbs of excess and sufficiency. These are words like too and enough that express an excess or sufficiency of a quality of a subject that is described by a polar word. They are similar to our presupposition shifter but rather than negating, these words impose a
polarity themselves and remove the otherwise polar content of the polar words they modify. Here is an example containing both kinds of adverbs along with negation:

- (-) The scroll button is overly $(x)$ sensitive $(+)$ at times; not $(g)$ sensitive $(+)$ enough $(x)$ at others.

The word overly imposes a negative polarity while enough imposes a positive polarity. Resolving precedence in the presence of other negating shifters however is beyond the scope of our current system and as this feature is sparse in our data, we omit this shifter presently. It is however another construct deserving further investigation.

There are also a number of failures due to the construct known as hedging. This is where the impact of an evaluation is reduced by also making a contrary statement with slightly less evaluative force. This can be noted in the following examples.

- (-) It doesn't have firewire, not(g) a real complaint(-) since most windows users don't generally have firewire cards themselves.
- (-) It could be a little bit bigger, but it's easy(+) to get used to.

Further experiments measuring strength of opinion in addition to direction could benefit from modeling this construct.

### 4.6 Remaining Language Features Affecting Polarity

Finally we are left with a number of classifiable errors and anomalies. These include language features such as irony and idiom, as well as complex usage and multiple opinions. The GI lexicon has a tag for idioms, and this could form the beginnings of an idiom database that could be used as a component in the polarity classification algorithm. The following are some examples:

- (-) 3/4 of the way through the first disk we played on it (naturally(+) 31 days after purchase) the dvd player froze.
- (+) I bought it for my trip to Buenos Aires, and also used it at the Iguazu Falls, and could not $(g)$ have asked for more perfect $(+)$ performance!
- (-) No games - it has a cool(+) screen - why not use it?
- (+) The small size is perfect $(+)$ for my little hands, but may perhaps be uncomfortable(-) or awkward(-) for a bigger person.

The first sentence has an ironic use of a positive word; the second an idiomatic use of negation that doesn't invert polarity. The third uses a complex structure that acts like an unmarked contrastive. The last sentence expresses multiple conflicting opinions.

### 4.7 Default Scoring Algorithms

The default score used by Hu-Liu is the previous sentence's score for a corpus consisting of multi-sentence reviews. To improve the cases where the default score is used, we explore the combination of our algorithm with the real-number score generating collocation-based log-likelihood algorithm of Yu-Hatzivassiloglou [YH03]. This proves to be ineffective for the highly refined augmented lexicon, since the previous sentence default accuracy increases as the nonzero accuracy increases, and this default outperforms the relatively noisy collocation scoring in this case. Nevertheless the collocation algorithm provides a better than random default scoring method for corpora of isolated sentences where the previous sentence default does not apply.

### 4.8 Application to Another Domain: Movie Reviews

We next apply our algorithm to another larger annotated sentence-level polarity test corpus (that of Pang et al. $[\mathrm{P}+02]$ ) in the new domain of movie reviews. Improvements are shown for most of our developed lexicons and discovered shifter features. We note that there are some important differences between the Hu-Liu corpus and that of Pang et al. One is that since the sentences are all isolated movie review summaries and not sequential sentences about the same subject, we cannot use the previous sentence's score as our default score. Also, since the sentence polarity is not focused around particular feature words, we cannot use something like the effective opinion tie breaking scheme of Hu and Liu. Since there are an equal number of positive and negative sentences in the Pang corpus, the best default score we can give is that of the most frequent class, resulting in 50\% accuracy in these cases.

We begin with the Hu-Liu lexicon with WordNet expansion. Results are shown in Table 21.

| Table 21: Hu Liu Lexicon with WordNet on Pang Corpus |  |  |  |  |
| :--- | :--- | :--- | :--- | ---: |
| Accuracies <br> Configuration <br>  <br>  <br>  <br> hulexWN -j <br>  <br> Total | $5746 / 10662$ | 0.5389 | $5771 / 10662$ | $\mathbf{l}$ |
|  |  |  |  |  |
|  | $4266 / 7702$ | 0.5539 | $4291 / 7702$ | 0.5571 |
|  | $3635 / 6440$ | 0.5644 | $3649 / 6419$ | 0.5685 |
| Zero | $631 / 1262$ | 0.5000 | $642 / 1283$ | 0.5004 |
| Null | $1480 / 2960$ | 0.5000 | $1480 / 2960$ | 0.5000 |
| Coverage |  |  |  |  |
| Nonnull | $7702 / 10662$ | 0.7224 |  |  |
| Nonzero | $6440 / 10662$ | 0.6040 | $6419 / 10662$ | 0.6020 |

We then try the HM lexicon with and without WordNet expansion. This dramatically improves accuracy from 0.541 to 0.613 . Interestingly, WordNet expansion improves both accuracy ( 0.602 to 0.613 ) and coverage ( 0.621 to 0.792 ) in this corpus. These results are found in Table 22.

| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | hmWN -j |  | hmWN -jdg |  | $\mathrm{hm}-\mathrm{j}$ |  | hm -jdg |  |
|  |  |  |  |  |  |  |  |  |
| Total | 6465/10662 | 0.6064 | 6544/10662 | 0.6138 | 6373/10662 | 0.5977 | 6418/10662 | 0.6020 |
| Nonnull | 5357/8445 | 0.6343 | 5436/8445 | 0.6437 | 4350/6617 | 0.6574 | 4395/6617 | 0.6642 |
| Nonzero | 4655/7041 | 0.6611 | 4700/6973 | 0.6740 | 3950/5817 | 0.6790 | 3965/5759 | 0.6885 |
| Zero | 702/1404 | 0.5000 | 736/1472 | 0.5000 | 400/800 | 0.5000 | 430/858 | 0.5012 |
| Null | 1108/2217 | 0.4998 | 1108/2217 | 0.4998 | 2023/4045 | 0.5001 | 2023/4045 | 0.5001 |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 8445/10662 | 0.7921 |  |  | 6617/10662 | 0.6206 |  |  |
| Nonzero | 7041/10662 | 0.6604 | 6973/10662 | 0.6540 | 5817/10662 | 0.5456 | 5759/10662 | 0.5401 |

Next we study the behavior of including non-adjective polar words by testing the combined and robust lexicons. While the combined lexicon does not show cumulative improvement with each part of speech, the robust lexicon does (from 0.608 to 0.614 to 0.617 to 0.620 ), as seen in Table 23.

| Table 23: Part of Speech Analysis in Combined and Robust Lexicons |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Accuracies | Configuration |  |  |  |  |  |  |  |  |  |  | $\mathrm{hx}-\mathrm{jn}$ | $\mathrm{hx}-\mathrm{jnr}$ |  | $\mathrm{hx}-\mathrm{jnrv}$ |  |
|  | $\mathrm{hx}-\mathrm{j}$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $6462 / 10662$ | 0.6061 | $6521 / 10662$ | 0.6116 | $6467 / 10662$ | 0.6066 | $6499 / 10662$ | 0.6096 |  |  |  |  |  |  |  |  |
|  | $4946 / 7632$ | 0.6481 | $5676 / 8972$ | 0.6326 | $5756 / 9240$ | 0.6229 | $5930 / 9524$ | 0.6226 |  |  |  |  |  |  |  |  |
| Nonzero | $4422 / 6586$ | 0.6714 | $4963 / 7546$ | 0.6577 | $5025 / 7780$ | 0.6459 | $5138 / 7942$ | 0.6469 |  |  |  |  |  |  |  |  |
| Zero | $524 / 1046$ | 0.5010 | $713 / 1426$ | 0.5000 | $731 / 1460$ | 0.5007 | $792 / 1582$ | 0.5006 |  |  |  |  |  |  |  |  |
| Null | $1516 / 3030$ | 0.5003 | $845 / 1690$ | 0.5000 | $711 / 1422$ | 0.5000 | $569 / 1138$ | 0.5000 |  |  |  |  |  |  |  |  |
| Coverage |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Nonnull | $7632 / 10662$ | 0.7158 | $8972 / 10662$ | 0.8415 | $9240 / 10662$ | 0.8666 | $9524 / 10662$ | 0.8933 |  |  |  |  |  |  |  |  |
| Nonzero | $6586 / 10662$ | 0.6177 | $7546 / 10662$ | 0.7078 | $7780 / 10662$ | 0.7297 | $7942 / 10662$ | 0.7449 |  |  |  |  |  |  |  |  |


| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | myhx - j |  | myhx -jn |  | myhx -jnr |  | myhx -jnrv |  |
|  |  |  |  |  |  |  |  |  |
| Total | 6478/10662 | 0.6076 | 6548/10662 | 0.6141 | 6574/10662 | 0.6166 | 6606/10662 | 0.6196 |
| Nonnull | 4636/6977 | 0.6645 | 5448/8462 | 0.6438 | 5545/8603 | 0.6445 | 5736/8923 | 0.6428 |
| Nonzero | 4208/6122 | 0.6874 | 4829/7224 | 0.6685 | 4900/7313 | 0.6700 | 5044/7541 | 0.6689 |
| Zero | 428/855 | 0.5006 | 619/1238 | 0.5000 | 645/1290 | 0.5000 | 692/1382 | 0.5007 |
| Null | 1842/3685 | 0.4999 | 1100/2200 | 0.5000 | 1029/2059 | 0.4998 | 870/1739 | 0.5003 |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 6977/10662 | 0.6544 | 8462/10662 | 0.7937 | 8603/10662 | 0.8069 | 8923/10662 | 0.8369 |
| Nonzero | 6122/10662 | 0.5742 | 7224/10662 | 0.6776 | 7313/10662 | 0.6859 | 7541/10662 | 0.7073 |

We then examine the negation shifter using the robust lexicon and these results are reported in Table 24.

| Table 24: Robust Lexicon with Negation Shifter |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- |
| Accuracies | Configuration |  |  |  |
|  | myhx -jdg |  |  |  |

We move on to the augmented lexicon and find that it extends to some degree to this new domain by showing further improved accuracies overall. Likewise, we find that the rest of our shifters improve accuracies in isolation and in conjunction. Since WordNet was found to be effective in this corpus, we expand the augmented lexicon and obtain still more accurate results. These results are found in Tables 25, 26, and 27.

| Table 25: Augmented Lexicon on Pang |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :---: |
| Accuracies | Configuration |  |  | myhx + -jnrvdg |  |
|  | myhx+-jnrv |  |  |  |  |
|  | $6611 / 10662$ | 0.6201 | $6668 / 10662$ | 0.6254 |  |
|  | $5757 / 8955$ | 0.6429 | $5814 / 8955$ | 0.6493 |  |
| Nonzero | $5065 / 7570$ | 0.6691 | $5091 / 7509$ | 0.6780 |  |
| Zero | $692 / 1385$ | 0.4996 | $723 / 1446$ | 0.5000 |  |
| Null | $854 / 1707$ | 0.5003 | $854 / 1707$ | 0.5003 |  |
| Coverage |  |  |  |  |  |
| Nonnull | $8955 / 10662$ | 0.8399 |  |  |  |
| Nonzero | $7570 / 10662$ | 0.7100 | $7509 / 10662$ | 0.7043 |  |


| Accuracies | Configuration |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | myhx+-jnrvdgc |  | myhx+-jnrvdgm |  | myhx+-jnrvdgp |  | myhx+-jnrvdgcmp |  |
|  |  |  |  |  |  |  |  |  |
| Total | 6693/10662 | 0.6277 | 6680/10662 | 0.6265 | 6674/10662 | 0.6260 | 6715/10662 | 0.6298 |
| Nonnull | 5839/8955 | 0.6520 | 5826/8955 | 0.6506 | 5818/8949 | 0.6501 | 5859/8949 | 0.6547 |
| Nonzero | 5117/7511 | 0.6813 | 5100/7502 | 0.6798 | 5095/7502 | 0.6792 | 5135/7502 | 0.6845 |
| Zero | 722/1444 | 0.5000 | 726/1453 | 0.4997 | 723/1447 | 0.4997 | 724/1447 | 0.5004 |
| Null | 854/1707 | 0.5003 | 854/1707 | 0.5003 | 856/1713 | 0.4997 | 856/1713 | 0.4997 |
| Coverage |  |  |  |  |  |  |  |  |
| Nonnull | 8955/10662 | 0.8399 | 8955/10662 | 0.8399 | 8949/10662 | 0.8393 | 8949/10662 | 0.8393 |
| Nonzero | 7511/10662 | 0.7045 | 7502/10662 | 0.7036 | 7502/10662 | 0.7036 | 7502/10662 | 0.7036 |


| Table 27: MyHX+ with Shifters and WordNet Expansion |  |  |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :---: |
| Accuracies | Configuration |  |  |  |  |  |  |
|  | myxx+WN -jnrv | myhx+WN -jnrvdg |  |  |  | myhx+WN -jnrvdgcmp |  |
|  |  |  |  |  |  |  |  |
|  | $6633 / 10662$ | 0.6221 | $6741 / 10662$ | 0.6323 | $6783 / 10662$ | 0.6362 |  |
| Nonnull | $6174 / 9743$ | 0.6337 | $6282 / 9743$ | 0.6448 | $6321 / 9739$ | 0.6490 |  |
| Nonzero | $5316 / 8027$ | 0.6623 | $5390 / 7960$ | 0.6771 | $5438 / 7972$ | 0.6821 |  |
| Zero | $858 / 1716$ | 0.5000 | $892 / 1783$ | 0.5003 | $883 / 1767$ | 0.4997 |  |
| Null | $459 / 919$ | 0.4995 | $459 / 919$ | 0.4995 | $462 / 923$ | 0.5005 |  |
| Coverage |  |  |  |  |  |  |  |
| Nonnull | $9743 / 10662$ | 0.9138 | $9743 / 10662$ | 0.9138 | $9739 / 10662$ | 0.9134 |  |
| Nonzero | $8027 / 10662$ | 0.7529 | $7960 / 10662$ | 0.7466 | $7972 / 10662$ | 0.7477 |  |

To improve coverage and handle zero and null scored sentences more effectively, we expand our lexicon further using the log-likelihood scoring algorithm. This leads to a further gain in overall accuracy (to 0.665), as reported in Table 28.

Table 28: MyHX+ with Shifters, WordNet, and Log-Likelihood Expansion

| Accuracies | Configuration |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | ---: |
|  | myhx+WNL-jnrv | myhx+WNL -jnrvdg |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Total | $7064 / 10662$ | 0.6625 | $7083 / 10662$ | 0.6643 | $7089 / 10662$ | 0.6649 |
| Nonnull | $7061 / 10657$ | 0.6626 | $7080 / 10657$ | 0.6644 | $7086 / 10657$ | 0.6649 |
| Nonzero | $7056 / 10646$ | 0.6628 | $7074 / 10646$ | 0.6645 | $7081 / 10647$ | 0.6651 |
| Zero | $5 / 11$ | 0.4546 | $6 / 11$ | 0.5455 | $5 / 10$ | 0.5000 |
| Null | $3 / 5$ | 0.6000 | $3 / 5$ | 0.6000 | $3 / 5$ | 0.6000 |
| Coverage |  |  |  |  |  |  |
| Nonnull | $10657 / 10662$ | 0.9995 | $10657 / 10662$ | 0.9995 | $10657 / 10662$ | 0.9995 |
| Nonzero | $10646 / 10662$ | 0.9985 | $10646 / 10662$ | 0.9985 | $10647 / 10662$ | 0.9986 |

We have found that our accuracy increased with each of our various improvements. It went from 0.54 to 0.61 with the change to using the intelligent lexicon from the simple seed set that was expanded by WordNet. Accuracy further increased to 0.63 with the addition of the other parts of speech to the lexicon along with some refinements. It rose to 0.64 with the application of the other shifters along with WordNet expansion. Finally it reached 0.66 with the application of the log-likelihood expansion.

### 4.9 Discussion of Results

In this section we summarize main steps that led to improvements in the algorithm. We also discuss things that failed to improve accuracy like WordNet for the Hu-Liu corpus, effective opinion tie breaking in some cases, and but-conjunction for effective opinion in some cases.

Through the process of failure analysis, we iteratively discover the next most important feature to model after each subsequent improvement. Since each step involves dealing with the next most common source of failure, we make the largest jumps in accuracy first, and then find more and more subtle refinements. The first important finding was that each of the pieces of the original Hu -Liu algorithm was quite sensitive to the initial seed set and the full lexicon used to score the sentences. The WordNet expansion algorithm as it was described added a lot of noise to the polarity word set and hence introduced a lot of errors. Also the effective opinion tie breaking was inconsistent as it helped in some product domains and not others, and overall was often less accurate than the previous sentence default.

Moving to the intelligent HM lexicon reduced the noise in the data and increased accuracy, and led to the next important finding. This was the validation that nouns, adverbs and verbs are important carriers of polarity in addition to adjectives. Adding those words from the GI lexicon increased coverage and accuracy, and after pruning the
list and adding to it in the next two phases of analysis, we were able to expose a range of more subtle contextual constructs used in the expression of opinions. There are a number of contextual valence shifters used to modify the polarity inherent in the lexical items of a sentence. A number of these types of shifters, such as negation, contrastives, modals, presuppositions, and but-conjunctions, invert the polarity of the lexical items they operate on, sometimes inverting the same word's polarity multiple times when they are present together. There are still other shifters like adverbs of excess and sufficiency and hedging that force polarity one way or the other, or reduce the strength of lexical polarity. For each of our main improvements to the algorithm, including lexicon refinement, parts of speech expansion, and contextual valence shifter adjustment, we were able to show a statistically significant increase in accuracy.

Like the Hu corpus, the Pang corpus shows increasing accuracy of prediction as we improve the lexicon and apply contextual valence shifters. There are however notable differences. One example is the relative domain dependence of lexicon words. A word like "complex" is generally negative for consumer products like cameras and dvd players but positive for movie plots and themes. This suggests polar words exist on a scale of broadness of application, where words like "annoying" and "enjoyable" are more universally applicable, while words like "complex" and "simple" have domain dependent polarity.

Because there are always cases where the predicted score is null or zero, default scoring algorithms are important. Effective opinion tie breaking is possible when there are feature words in the corpus but is found to be only marginally effective. Using the score of the previous sentence when the corpus consists of paragraphs of evaluative text rather than isolated sentences works well and is best with a refined intelligent lexicon. Default scoring using log-likelihood lexical scores for non-seed words based on their collocation frequency with seed words is the next best algorithm and the only option for isolated sentence corpora like the Pang corpus.

## 5 Conclusions and Further Work

Understanding evaluative language is an important task in natural language processing with widespread applications. We have focused on the task of sentence level opinion classification in the hope of discovering and classifying the basic building blocks of evaluative discourse. Starting with a collection of existing algorithms, we have built a flexible framework capable of classifying opinions with accuracy exceeding current state-of-the-art results. We have discovered many of the important factors in performing opinion classification, including the parts of speech that carry polarity, the range of polarity attached to the different uses and senses of words, and some of the linguistic constructs that shift the polarity contribution that words make to a sentence.

There are a number of other algorithms and ideas suggested in the literature that were not applied to sentence level polarity prediction but that we feel could be integrated into our system as improvement tools. One technique is the semantic orientation equation (called SO-A, semantic orientation by association) of Turney and Littman [TL03]. This is based on the idea of pointwise mutual information, and works in a way very analogous to the log-likelihood equation of Yu and Hatzivassiloglou [YH03]. It however uses web search hits with a proximity operator instead of corpus collocation to compute a polarity score. The work was originally done using AltaVista's NEAR operator, but that operator has since become defunct (upon Yahoo's acquisition of AltaVista in 2003). Turney has since developed a Beowulf cluster containing a database of a terabyte of web pages and an associated query language that includes a proximity operator. So SO-A can now be computed using this static approximation of the internet. This could be used as an alternate method for computing lexical polarity scores, or could be used as a method to cross-check or filter out scores computed by other methods.

To improve the accuracy produced by the Yu and Hatzivassiloglou log-likelihood collocation algorithm, we would like to try augmenting our corpus with other datasets to increase the amount of collocation that will occur. This should produce a larger number of more accurate scored polarity words.

The work of Riloff and her colleagues $[R+03]$ suggests a method for extracting linguistic patterns automatically. They have used a number of unsupervised machine learning algorithms, such as AutoSlog-TS, MetaBoot, and Basilisk that implement the technique they call "extraction pattern bootstrapping". They use these algorithms to discover patterns indicative of subjectivity, but we would like to explore the possibility of applying these or similar techniques to discovering patterns to be used in polarity prediction. These might include patterns that carry polarity or ones that shift the polarity of the constituent polar lexical items. We will attempt to classify them based on linguistic phenomena, or possibly use them to propose new linguistic models of evaluative language.

To evaluate the patterns, we will explore the idea of measuring their consistency and strength using metrics of polarity like the log-likelihood and SO-A equations. One idea for doing this is to use the SO-A equation on a series of $n$-grams constructed by taking a given pattern and substituting in different lexicon words. By doing this we would like to be able to rank the patterns by consistency (how consistently they shift the polarity in the same direction) and strength (how far they shift the polarity).

To optimize the way we use our various lexical and contextual classifiers, we would like to experiment with boosting. Specifying our polarity features as classifiers for AdaBoost [FS96] would allow us to optimize how they are used on our dataset for classification.

We provide a first pass of various shifters in our implementation and prove their mechanics but the sets of marker words and phrases in each shifter category are subject to expansion. This could be achieved through further rounds of failure analysis on new corpora from other domains.

We would also like to investigate but-conjunction, adverbs of excess and sufficiency, and hedging in more detail. But-conjunction is a common form in evaluative text so finding ways to distinguish its shifter usage would be very helpful. Modeling adverbs of excess and sufficiency and hedging would add new layers to the system involving precedence and strength that could improve accuracies further. Finally, we would like to investigate more complex rhetorical devices such as idiom and irony. Starting with the General Inquirer, we would like to compile a database of idiomatic patterns that can effect polarity, and add this component to our system.

We have established a framework of tools and algorithms for opinion classification and a methodology for expanding this framework. With further research, the precision of this system will continue to grow as will its applicability to the task of understanding evaluative text.

## References

[BW99] Rebbeca Bruce and Janyce Wiebe. 1999. Decomposable modeling in natural language processing. Computational Linguistics, 25(2).
[CV04] Rudy Cilibrasi and Paul Vitanyi. 2004. Automatic meaning discovery using Google. http://xxx.lanl.gov/abs/cs.CL/0412098.
[C95] Paul Cohen. 1995. Empirical Methods for Artificial Intelligence. MIT Press, Cambridge, MA.
[E+00] Gerard Escudero, Lluis Marquez, and German Rigau. 2000. Boosting applied to word sense disambiguation. In Proceedings of the 12th European Conference on Machine Learning, pages 129-141.
[FS96] Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, pages 148-156.
[G+03] Andrew Gordon, Abe Kazemzadeh, Anish Nair, and Milena Petrova. 2003. Recognizing expressions of commonsense psychology in English text. In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03), pages 208-215.
[HM97] Vasileios Hatzivassiloglou and Kathy McKeown. 1997. Predicting the semantic orientation of adjectives. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97), pages 174-181.
[HL04] Hu, M., and Liu, B. 2004. Mining Opinion Features in Customer Reviews. In Proceedings of the 21st National Conference on Artificial Intelligence (AAAI2004).
[HT00] Susan Hunston and Geoff Thompson. 2000. (Eds.) Evaluation in Text: Authorial Stance and the Construction of Discourse. Oxford University Press.
[JM00] Daniel Jurafsky and James H. Martin. 2000. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall.
[M00] Lluis Marquez. 2000. Machine learning and natural language processing. LSI-00-45-R. Departament de Llenguatges i Sistemes Informatics (LSI), Universitat Politecnica de Catalunya (UPC). Barcelona, Spain.
[M03] J. R. Martin. 2003. Introduction. Text 23(2). pages 171-181. Walter de Gruyter.
[M+90] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An on-line lexical database. International Journal of Lexicography, 3(4):235-244.
[RM98] Rosamund Moon. 1998. Fixed Expressions and Idioms in English: A CorpusBased Approach. Oxford University Press.
[N00] NLProcessor - Text Analysis Toolkit. 2000. http://www.infogistics.com/textanalysis.html
[ $\mathrm{P}+02]$ Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002), pages 79-86.
[PZO4] Livia Polanyi and Annie Zaenen. 2004. Contextual valence shifters. In Proceedings of the 2004 AAAI Spring Symposium - Exploring Attitude and Affect in Text: Theories and Applications, pages 114-119.
[R+03] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In Proceedings of the 7th Conference on Natural Language Learning (CoNLL-2003), pages 25-32.
[S02] Robert E. Schapire. 2002. The boosting approach to machine learning: an overview. In Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA.
[S04] Stanford Natural Language Processing Group. 2004. Stanford Tagger. http://nlp.stanford.edu/software/tagger.shtml
[S+66] Philip J. Stone, Dexter C. Dunphy, Marschall S. Smith, and Daniel M. Ogilvie. 1966. The General Inquirer: a computer approach to content analysis. M.I.T. studies in comparative politics. MIT Press, Cambridge MA.
[TH00] Geoff Thompson and Susan Hunston. 2000. Evaluation: An Introduction. In Susan Hunston and Geoff Thompson (Eds.), Evaluation in Text: Authorial Stance and the Construction of Discourse, pages 1-27. Oxford University Press.
[TL03] P. Turney and M. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. ACM Transactions on Information Systems (TOIS), 21(4):315-346.
[W00] Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), pages 735-740.
[W+01] Janyce Wiebe, Rebecca Bruce, Matthew Bell, Melanie Martin, and Theresa Wilson. 2001. A corpus study of evaluative and speculative language. In Proceedings of the 2nd ACL SIGdial Workshop on Discourse and Dialogue. 2001.
[W+99] J. Wiebe, R. Bruce, and T. O'Hara. 1999. Development and use of a gold standard data set for subjectivity classifications. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99), pages 246--253, University of Maryland.
[WW03]Theresa Wilson and Janyce Wiebe. 2003. Annotating opinions in the world press. In Proceedings of the 4th ACL SIGdial Workshop on Discourse and Dialogue (SIGdial-03), pages 13-22.
[YN04] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. 2003. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM-2003).
[YH03] Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003), pages 129-136.

## Appendix - Project Code

These scripts correspond to the framework presented in Chapter 3. There is one preprocessing script for for the NLProcessor (0-hu-preproc.pl) and there are two post processing scripts for the Standford Tagger (0-pang-prepend-stanford.pl followed by 0-post-stanford.pl). The tagged xml corpus is then formatted with 1-proc-corp.pl. At this point there is the option to expand the lexicon using the WordNet algorithm (scripts wninit.pl, 2-wn-extract-adj.pl, and 3-wn-orient-pred.pl) or the log likelihood algorithm (scripts 2-log-lex-count.pl, 3-log-word-stats.pl, and 4-log-word-scores.pl). Finally, the formatted corpus is scored using 5 -score-corp.pl (possibly across multiple corpora with run-5-score-corp.pl). Further usage notes are found in the script comments.

```
==
===
#!/home/bin/per1
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: 0-hu-preproc.pl
###author: Adam Longton, longton@cs.ubc.ca
#******************************************************************
#description, inputs and output:
#takes a corpus file from the 'customer review data' of Hu and Liu '04.
#converts from native format to xml markup, for input to the NLProcessor
#pos-tagger and chunker
#*****************************************************************
use strict;
use warnings;
my $usage = "usage: perl $0 <corpusfile>\n";
die $usage if ($#ARGV < 0);
my $line;
#get input file
my $corpfile = shift;
open(CORP, $corpfile) or die "file error: $usage";
print "<FILE>\n";
while ($line = <CORP>){
    chomp $line;
    #clean up ampersands and less-thans to be valid xml input
    $line =~ s/&([^\#])/&\#38;$1/g;
    #$line =~ s/</&\#60;/g; --don't need cuz not in corpus
    if ( $line =~ /(.*)\#\#(.+)/ ){ #a sentence
    print "<S";
    if ($1 eq ""){print ">$2</S>\n";}
    else {print " info=\"$1\">$2</S>\n";}
        }
        elsif ( $line =~ /^\[t\] ?(.*)/ ){ #a review title
    print "<TITLE>$1</TITLE>\n";
    }
    #else do nothing, just drop it
}
print "</FILE>\n";
close(CORP);
###END
===
#!/home/bin/perl
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: 0-pang-prepend-stanford.pl
###author: Adam Longton, longton@cs.ubc.ca
#*******************************************************************
#description, inputs and output:
#takes a corpus file from the pang+lee movie review snippet/sentence polarity
#corpus and prepends it with score[+1]## (or score[-1]##) for input into 0-post-stanford
#script.
#*********************************************************************
use strict;
use warnings;
my $usage = "usage: per1 $0 <corpusfile>\n";
die $usage if ($#ARGV < 0);
my $line;
#get input file
my $corpfile = shift;
open(CORP, $corpfile) or die "file error: $usage";
while ($line = <CORP>) {
    chomp $line;
```

```
    #clean up ampersands and less-thans to be valid xml input
    #$line =~ s/&([^\#])/&\#38;$1/g;
    #$line =~ s/</&\#60;/g; #not in corpus
    #for pang, rather than xml codes, use own symbols:
    $line =~ s/\</LT/g;
    $line =~ s/\>/GT/g;
    #could also remove non UTF-8 chars here by char ascii # limit maybe
    print "score[+1]##$line\n"; # for positive sentences
    #print "score[-1]##$line\n"; #for negative setences
}
close(CORP);
```


## \#\#END

## ==

```
\#!/home/bin/perl
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: 0-post-stanford.pl
###author: Adam Longton, longton@cs.ubc.ca
#*****************************************************************
#description, inputs and output:
#takes a corpus file in the format of the 'customer review data' of Hu and Liu '04 that
has been
#stanford pos-tagged.
#converts from native plus stanford pos-tagged with '/' separators to
#xml markup (with '_' separators), for input to phase 1 script.
#*****************************************************************
use strict;
use warnings;
my $usage = "usage: perl $0 <corpusfile>\n";
die $usage if ($#ARGV < 0);
my $line;
#get input file
my $corpfile = shift;
open(CORP, $corpfile) or die "file error: $usage";
#derive corp name from filename:
$corpfile =~ /([^\/]+)\-tagged\.txt/;
my $corp = $1;
my $sid = 0;
my $tid = 0;
my $info;
my $sent;
print "<FILE fid=\"$corp\">\n";
while ($line = <CORP>){
    chomp $line;
    #don't clean up ampersands and less-thans to be valid xml input
    #cuz already done in preproc
    #$line =~ s/&([^\#])/&\#38;$1/g;
    #$line =~ s/</&\#60;/g; --not in corpus
    if ( $line =~ /(.*)\#\#(.+)/ ){ #a sentence
        $info = $1;
        $sent = $2;
        $sid++;
        #switch / to _ (note that it assumes lines end w/ a space)
        $sent =~ s/\/([^\/]+) /_$1 /g;
        print "<S sid=\"$corp:$sid\"n;
        if ($info eq ""){print ">$sent</S>\n";}
        else {print " info=\"$infol">$sent</S>\n";}
    }
    elsif ( $line =~ /^\[t\] ?(.*)/ ){ #a review title
        $tid++;
        print "<TITLE tid=\"$corp:$tid\">$1</TITLE>\n";
```

```
    }
    #else do nothing, just drop it
    else{
    print STDERR "$line\n";
    }
}
print "</FILE>\n";
close (CORP);
##END
===
#!/home/bin/perl
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: l-proc-corp.pl
###author: Adam Longton, longton@cs.ubc.ca
#*****************************************************************
#description, inputs and output:
#
#this script is phase 1 of a multi-phase architecture that implements
#the sentence-level sentiment-polarity calculating scheme of
#[Hu and Liu 2004]
#
#input: a corpus of POS-tagged sentences. more precisely, a file from
#hu and liu's 'customer review data' corpus 2004, preprocessed by
#0-hu-preproc.pl to convert it into XML, and pos-tagging being done
#by NLProcessor. (post proc'd for sent ids too i think, only so postag
#doesnt need be done again.)
#
#the command to pos-tag it looks like, for the apex.xml corpus,
#
#cat ../../hu-corpus/apex.xm1 | bin/n1p.sh -q ".*/FILE" -qs ".*/S"
#-xml_flat -show_tags | bin/sgdelmarkup -q ".*/W" -p "\{#\}_\{C\}"
# > apex-tagged.xml
#
#the tagset is a modified penn treebank set. this script generalizes them
#in this way: N*->N, V*->V, J*->J, and R*->R unless it's RP (particle).
#
#output: to standard output, each line is a clause, of form:
#<ann>,0,0(,<stem>:<pos>)*
#that is, a polarity annotation <ann> which can have the values
#+ for only positive, - for only negative, and m for mixed opinions, or
#else a 0 for unannotated and presumably neutral opinion. this assumption
#is not always true tho because they are subject targeted.. if a positive
#sentiment is being expressed but not toward any particular feature then
#it might be lablled neutral but positive words will want to score it
#positively. an example is the sentence 'awesome_JJ !_.' found
#in the corpus.
# we also have two more fields: sentence id and
#graded score. they are thrown into subfields of field 0 sep'd by semicolons.
#other fields continued:
#2 intially zero slots for pstv and ngtv
#lexicon-word counts (to be filled in later if using log-likelihood), and the word:pos
pairs
#for that clause.
#There are option-flags for choosing which pos's you want to filter for.
#-n -v -j -r for noun verb adjective adverb, respectively.
#
#example:
#
#input line from cat.xml:
#<S sid="cat:23" info="cat[+2], dog[+3]">The_DT cat_NN and_CC
#especially_RB the_DT dog_NN are_VB very_RB very_RB good_JJ ._. </S>
#
#command-line execution:
#>1-hu-proc-corp.pl -vjr cat.xml
```

```
#
#output line:
#cat:23;+;2.5,0,0, especially:R,are:V,very:R,very:R,good:J
#******************************************************************
use strict;
use warnings;
my $usage = "usage: perl $0 <xmlcorpusfile>\n";
die $usage if ($#ARGV < 0);
#process infile
my $corpfile = shift;
open(CORP, $corpfile) or die "file error: $usage";
#variables
my $line; my @lineArr; my $sent; my @outArr;
my $word; my $pos; my $pair;
my $info; my @infoArr;
my @annArr; my $sum; my $n;
# eff opinion - need the info field for ph5 scoring
my @infoOutArr;
my $number = 0; my $rounded = 0;
#proc file - line in line out..
while($line = <CORP>){
    if( $line =~ /<S[^>]* info=\"([^\">]+)\"/ ){ #if an annotated sent,
        $info = $1;
        @outArr= (0,0,0);
        @annArr = (0,0,0);
            #process binary annotation
        if( $info =~ 八\\\+/ ){
            if( $info =~ 八[-/ ){ #mixed
            $annArr[1]='m';
            }
            else{ $annArr[1]='+'; } #positive
        }
        elsif( $info =~ /\[-/ ) { #negative
            $annArr[1]='-';
        } # other option is to take sign of graded ann
        #only proceed for + and - (ignore m and 0, 0 including the two
        #cases where no score provided by hu-liu..
        if( $annArr[1] eq '+' || $annArr[1] eq '-' ){
            if( $line =~ /<S[^>]*>\s*([^<]*)<\/S>/ ){
                    $sent = $1;
                    ##do delimiter switching here:***************
                    $sent =~ s/\,/CM/g;
                    $sent =~ s/\:/CL/g;
                    $sent =~ s/\;/SC/g;
                    @lineArr = split / /,$sent;
            #process other xml attrib (sid)
            if( $line =~ /<S[^>]* sid=\"([^\">]+)\"/ ){
                $annArr[0] = $1;
            }
            #also put the info attrib into the output, for eff opinion scoring in ph5.
            # split on commas, remove leading spaces, replace other whitespace with an
            # underscore, join with colons. clean up that occurence of ";" in
            # anomalous "at&#38;t" feature word.
            @infoOutArr = split /,/,$info;
            for(my $i = 0; $i <= $#infoOutArr; $i++ ) {
                    #remove leading spaces
                    $infoOutArr[$i] =~ s/^\s+//;
                    #replace other whitespace with underscore
                    $infoOutArr[$i] =~ s/\s+/_/g;
                    #put "&"'s back in
```

```
            $infoOutArr[$i] =~ s/&#38;/&/g;
            #otherwise should remove any remaining semicolons, but unnecessary for
huliu corpus
    }
    my $infoOutStr = join ':',@infoOutArr;
    $outArr[2] = "FEATURES=" . $infoOutStr;
    #process graded annotation using info attribute
    @infoArr = split /\[/,$info;
    #if( $info !~ /^\[/ ){ <--for safety but unnecessary
    shift @infoArr; #pop off initial part of string
    #all other parts start with a number
    #}
    $n = 0;
    $sum = 0;
    foreach my $chunk ( @infoArr ) {
            if( substr($chunk,0,1) eq '+' ) {
                $sum += substr($chunk,1,l);
                $n++;
            }
            elsif( substr($chunk,0,l) eq '-' ){
                $sum -= substr($chunk,1,1);
                $n++;
            }
    }
    if( $n>0 ){
            $number = $sum/$n;
            $rounded = sprintf("%.3f", $number);
            $annArr[2] = $rounded;
    }
}
#insert annArr into field zero of outArr
$outArr[0] = join ';',MannArr;
#process sentence
foreach $pair (@lineArr) { #extract words of valid pos
    if($pair =~ /([^_]+)_([^_]+)/ ) { #disallows __ and _. cases
            $word = $l;
            $pos = $2;
            if( $pos =~ /^NN.*/ ) { #matches NN[SP(PS)]?
                #if($optsHash{'n'}){
                push @outArr, ($word . ":N");
            #}
            }
            elsif( $pos =~ /^VB.*/ ){ #matches VB[DGNPZ]?
                #if($optsHash{'v'}){
                push BoutArr, ($word . ":V");
                #}
            }
            elsif( $pos =~ /^JJ.*/ ){ #matches JJ[RS]?
                #if($optsHash{'j'}){
                push @outArr, ($word . ":J");
                #}
            }
            elsif( $pos =~ /^RB.*/ ) { #matches RB[RS]?
                #if($optsHash{'r'}) {
                push @outArr, ($word . ":R");
                #}
            }
            elsif( $pos =~ /^RP.*/ ) { #matches particle RP
                push @outArr, ($word . ":P");
            }
            else{
                push @outArr, ($word . ":$pos");
            }
    }
}
if($#outArr > 2){ ###print to outfile only if nonempty
    print join(',',QoutArr);
```

```
                print "\n";
                }
        }
    }
}
close(CORP);
###END
===
#!/home/bin/perl
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: wn-init.pl
###author: Adam Longton, longton@cs.ubc.ca
# this script inits some required environment variables for 3-wn-orient-pred.pl
# paths depend on particular experimental environment
`setenv WNHOME /CS/public/generic/lib/pkg/WordNet-2.1/`; #location of WordNet package
`setenv PERL5LIB /.autofs/homes/ubccshome/l/longton/adam/proj/wordnet/`; # location of
WordNet-QueryData-1.46 (WordNet perl interface)
#END
===
#!/home/bin/perl
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: 2-wn-extract-adj.pl
###author: Adamm Longton, longton@cs.ubc.ca
# this script is the first of two used to expand an adjective lexicon using wordnet. this
one extracts
# the adjectives from the corpus.
#*****************************************************************
#input: a corpus of stemmed and POS-tagged sentences, in the format
#of the output of 1-proc-corp.pl
#example:
#
#input line from lout.txt:
#cat:23;+;2.5,0,0, cats:N, are:V,nice:J,very:R,very:R, good:J, funny:J
#
#command-line execution:
#>2-hu-extract-adj.pl lout.txt > 2out-adjlist.txt
#
#output:
#funny:J
#good:J
#nice:J
#******************************************************************
use strict;
use warnings;
my $usage = "usage: perl $0 <corpusfile>\n";
die $usage if ($#ARGV < 0);
my $corpfile = shift;
open(CORP, $corpfile) or die "file error: $usage";
#process infile, output an adj-list
my $line;
my %adjHash = ();
my @lineArr; my $word;
my $adjCount = 0; my $wordCount = 0;
while ($line = <CORP>){
    chomp $line;
    @lineArr = split /,/,$line;
    for(my $i=3;$i<=$#1ineArr;$i++){
    $word = $lineArr[$i];
    $wordCount++;
    #throw adjectives into a hash
```

```
    #val=count for reference
    if( $word =~ 八:J/ ){
        $adjHash{$word}++;
        $adjCount++;
    }
    }
}
close(CORP);
my $disCount = scalar keys %adjHash;
#print adjs (for input into wordnet script (ph3))
foreach $word (sort keys %adjHash){
    print "$word\n";
}
print STDERR "Num Distinct Adjs = $disCount\n";
print STDERR "Num Adjs = $adjCount\n";
print STDERR "Num Words = $wordCount\n";
###END
===
#!/home/bin/perl
#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
###filename: 3-wn-orient-pred.pl
###author: Adam Longton, longton@cs.ubc.ca
# this script is the second of two used to expand an adjective lexicon using wordnet.
this one does the wordnet
# expnasion given the adjectives from the corpus.
#*****************************************************************
#input:pstv and ngtv word lexicons and an adj-1ist (2out).
#output: an (expanded) wordscore file. this means a list of adjs consisting
#Of those from pstv file given score +1, ngtv -1, and those adjs from the
#adjlist that were synonyms or antonyms of known psv and ngv words
#according to wordnet. in these cases a syn has the same score as its root and
#an ant the opposite (if +1 then -1 for eg).
#example:
#input:
#pv.txt:
#good:J
#quick:J
#nv.txt:
#bad:J
#2out-adjlist.txt:
#fast:J
#good:J
#slow:J
#nice:J
#nonwerd:J
#command-1ine execution:
#>3-wn-orient-pred.pl pv.txt nv.txt 2out-adjlist.txt > 3out.txt
#output:
#bad:J, -1
#fast:J,1
#good:J,1
#nice:J,1
#quick:J,1
#slow:J,-1
#*****************************************************************
#***setup+proc options**************************************************
use strict;
use warnings;
```

```
#wordnet stuff
use WordNet::QueryData;
my $wn = WordNet::QueryData->new;
```

```
my $usage = "usage: perl $0 [-sa#] <pstvfile> <ngtvfile> <adjfile>\n";
```

my \$usage = "usage: perl \$0 [-sa\#] <pstvfile> <ngtvfile> <adjfile>\n";
die $usage if ($\#ARGV < 2);
die $usage if ($\#ARGV < 2);
my %optsHash = ('s',0,'a',0,'n',0); \#flags, default all off
my %optsHash = ('s',0,'a',0,'n',0); \#flags, default all off
\#process options
\#process options

# -s : include 'similar to' words

# -s : include 'similar to' words

# -a : include 'see also' words

# -a : include 'see also' words

# -\#, where \# is 1-9 : search only senses 1-\#

# -\#, where \# is 1-9 : search only senses 1-\#

# -\# is stored as 'n', where value, rather than being just a boolean, is

# -\# is stored as 'n', where value, rather than being just a boolean, is

\#also a count value
\#also a count value
my \$optsStr = "";
my \$optsStr = "";
my \$opts;
my $opts;
my @optsArr;
my @optsArr;
if($\#ARGV > 2){
if(\$\#ARGV > 2){
\$opts = shift;
\$opts = shift;
\$optsStr = \$opts;
\$optsStr = \$opts;
\$opts =~ /^-/ or die "file error: \$usage";
\$opts =~ /^-/ or die "file error: $usage";
        if($opts eq '-'){die "file error: $usage";}
        if($opts eq '-'){die "file error: \$usage";}
$opts = substr($opts,1);
$opts = substr($opts,1);
if(\$opts =~ /[^sal-9]/){die "file error: $usage";}
        if($opts =~ /[^sal-9]/){die "file error: $usage";}
            @optsArr = split //,$opts;
@optsArr = split //,\$opts;
foreach my \$c (@optsArr){
foreach my \$c (@optsArr){
if( \$c =~ /[1-9]/ ) {
if( \$c =~ /[1-9]/ ) {
$optsHash{'n'}=$c;
$optsHash{'n'}=$c;
}
}
else{
else{
$optsHash{$c}=1;
$optsHash{$c}=1;
}
}
}
}
}
}
\#***file processing setup***************************************************
\#process infiles
my \$pfile = shift;
my \$nfile = shift;
my \$adjfile = shift;
open(PSTV, \$pfile) or die "file error: \$usage";
open(NGTV, \$nfile) or die "file error: \$usage";
open(ADJ, \$adjfile) or die "file error: \$usage";

# append the outputfiles with the same suffix found in the adjfile (ie which corpus was

it)
my \$corpTag = "n;
if( \$adjfile =~ /2out(-.+)\.txt/ ){
\$corpTag = $1;
}
open(SEEDY, ">seedyAdjs$corpTag.txt") or die "error trying to open filel for writing";
open(ADDED, ">addedAdjs$corpTag.txt") or die "error open file2 for writing";
open(SCORED,">scoredAdjs$corpTag.txt") or die "error open file3 for writing";
open(UNUSED, ">unusedSeeds\$corpTag.txt") or die "error trying to open filel for writing";

# put the command in the added file (which now also has the via words)

my \$commandAndArgs = "perl \$0 \$optsStr \$pfile \$nfile \$adjfile";
print ADDED "command: \$commandAndArgs\n";
\#hashes to hold seed words (pv + nv) and unscored adjs
my %seedHash = ();
my %adjHash = ();
\#hash to hold adjs found in orig seeds
my %seedyAdjs = ();
\#hash to hold adjs with synonyms in (current) seeds list
my %addedAdjs = ();
\#mar08 hash to hold which synonym of an added word caused it to be added
my %addedViaHash = ();
\#aug2- hold unused seeds
my %unusedSeeds = ();

```
```

\#hashes to hold already generated syns and ants arrays for words from adjlist
my %synHash = ();
my %antHash = ();
\#***file processing************************************************************
my $line;
while ($line = <PSTV>) {
chomp \$line;
$seedHash{$line}=1;
}
close(PSTV);
while (\$line = <NGTV>) {
chomp \$line;
$seedHash{$line}=-1;
}

# make a copy of orig seed hash for compare at end..

my %origSeedHash = %seedHash;
close (NGTV) ;
while (\$line = <ADJ> ) (
chomp \$line;
$adjHash{$line}=undef;
}
close(ADJ);
my @adjs = sort keys %adjHash;
my \$s = scalar @adjs;

# to see what's getting cut, what's getting used

print "", "START adjlist(N=\$s): ", join(", ", @adjs), "\n\n";
\#my step - prune seeds from adjs ******************************************
foreach my $seed (keys %seedHash) {
    if( exists($adjHash{$seed}) ) {
        delete($adjHash{\$seed});
\#but add it to holder hash with corrsp orientation
$seedyAdjs{$seed} = $seedHash{$seed};
}
else{
$unusedSeeds{$seed} = $seedHash{$seed};
}
}
my \$uCount = scalar keys %unusedSeeds;
print "\nunused seeds count: \$uCount\n";
foreach my $seed (sort keys %unusedSeeds)(
    print UNUSED "$seed,$unusedSeeds{$seed}\n";
}
@adjs = sort keys %adjHash;
$s = scalar @adjs;
print "". "PRUNED (seedless) adjlist(N=$s): ", join(", ", @adjs), "\n\n";
\#HL's algorithm***********************************************************
my \$size1; my \$size2;
my \$iter = 1;
do{
\$size1 = scalar keys %seedHash;
orientationSearch(\%adjHash, \%seedHash, \%addedAdjs, \%addedViaHash);
$size2 = scalar keys %seedHash;
    print n", "XX($iter)XX size1=$size1 and size2=$size2\n\n";
$iter++;
}while($size1 != \$size2);
print "", "\nEND (unused) adjlist(N=" , scalar keys %adjHash, "): ", join(", ", sort keys
%adjHash), "\n\n";
\#***output*******************************************************************
my \$sCount = scalar keys %seedyAdjs;
print "\nseedy adj count: \$sCount\n";
foreach my \$adj (sort keys %seedyAdjs){

```
```

    print SEEDY "$adj,$seedyAdjs{$adj}\n";
    }
my \$aCount = scalar keys %addedAdjs;
print "\nadded adj count: \$aCount\n";
foreach my $adj (sort keys %addedAdjs) {
    print ADDED "$adj,$addedAdjs{$adj},\t\t\t$addedViaHash{$adj}\n";
}
foreach my \$adj (keys %seedyAdjs){
$addedAdjs{$adj} = $seedyAdjs{$adj};
}
\$aCount = scalar keys %addedAdjs;
print "\ntotal scored adj count: \$acount\n\n";
\#print output that will be used for scoring:
foreach my $adj (sort keys %addedAdjs){
    print SCORED "$adj,$addedAdjs{$adj}\n";
}
\#\#\#END MAIN section********************************************************
\#***subroutines**************************************************************
\#note we prune first adjs that are seeds. then no need to
\#check for existence. instead, when one is added to seeds, remove it
\#from adjs.
sub orientationSearch{
my (\$adjRef, \$seedRef, \$addedRef, \$addedViaRef) = @_;
my @synonyms = ();
my @antonyms = ();
my \$word;
my \$found;
foreach my $adj (keys %{$adjRef}) {
$found = 0;
        #get synonyms
        @synonyms = getSynonyms($adj);
print "", "--grabbed synonyms of \$adj\n";
\#for each, try to find it in seedlist. if do, then add orig word
\#with same polarity, and remove it from adjlist.
LOOP: foreach my $syn (@synonyms) {
            if( exists($seedRef->{\$syn}) ) {
$seedRef-> {$adj}=$seedRef-> {$sym};
delete($adjRef-> {$adj});
\#and add it to added hash
$addedRef-> {$adj}=$seedRef-> {$sym};
\#mar08 and add it and its via to the addedVia hash
$addedViaRef->{$adj}="via syn: \$syn";
\$found = 1;
print "", "-0-0-0-0-added \$adj to seedslist via \$syn\n";
last LOOP;
}
}

```
    \#if not found yet, check antonyms.. if then, add w/ opp polarity.
        if( ! \$found ) \{
            \#get antonyms
            @antonyms = getAntonyms(\$adj);
            print "", "--grabbed antonyms of \$adj\n";
            LOOP: foreach my \$ant (eantonyms) \{
            if ( exists (\$seedRef->\{\$ant\}) )
                        \$seedRef->\{\$adj\}= (-1) * \$seedRef->\{\$ant\};
                    delete (\$adjRef->\{\$adj\});
                        \#add it to added hash
                        \$addedRef->\{\$adj\}= (-l) * \$seedRef-> \{\$ant\};
                        \#add it and its via to the addedVia hash
                        \$addedViaRef->\{\$adj\}="via anto: \$ant";
                        print "n, "-0-0-0-0-added \$adj to seedslist via \$ant\n";
                        last LOOP;
                \(\}\)
            \}
```

        }
    }
    print "", "\n";
    }
\#takes a mysymtax word like good:J,
\#turns it into a wordnet:querydata word, like good\#a. restrict in and out to
\#non-phrasal (single word) lexemes.
\#optimization: only generate syn(and ant) lists at most once
sub getSynonyms{
my ($adj) = a_;
    #unless we've already generated the synonyms on a prev iteration..
    unless( exists($synHash{\$adj}) ) {
my \$word;
my \$w;
my Osenses = ();
my %syns = ();
my @currSyms = ();
my @currSims = ();
my @currAlsos = ();
\#convert into wordnet syntax
\$adj =~ /([^\:]+)\:J/;
\$word = \$1 . "\#a";
\#generate all synonyms: use synset, 'similar to' synsets,
\#and 'see also' synsets.
@senses = $wn->querySense($word);
\#chop sense list down to 'n' values if that option is nonzero
if( \$optsHash{'n'} ) {
if( \$\#senses >= \$optsHash{'n'} ){
splice @senses, \$optsHash{'n'}; \#chop from n onward
}
}
foreach my \$sense (@senses) {
@currSyns = $wn->querySense($sense, "syns");
foreach my \$syn (@currSyns){
\$syn =~ /^([^\#]+)\#/; \#grab the word before the first '\#'
\$w = \$1;
\#if not multiword lex, add it
unless( \$w =~ /[_ ]/ ){ \#expect only _ for multis, but be safe
$syns{"$w:J"}=undef;
}
}
if( \$optsHash{'s'} ){
\&currSims = $wn->querySense($sense, "sim");
\#print " 'similar to' synsets:\n";
foreach my \$sim (@currsims) {
@currSyns = $wn->querySense($sim, "syns");
\#print " \$sim: ", join(", ", @currSyns), "\n";
foreach my \$syn (@currSyns){
\$syn =~ /^([^\#]+)\#/; \#grab word before first '\#'
\$w = \$1;
\#if not multiword lex, add it
unless( \$w =~ / [_ ]/ ) { \#expect only _, but be safe
$syns{"$w:J"}=undef;
}
}
}
}
if( \$optsHash{'a'} ){
@currAlsos = $wn->querySense($sense, "also");
\#print " 'see also' symsets:\n";
foreach my \$also (@currAlsos){
GcurrSyns = $wn->querySense($also, "syns");
\#print " \$also: ", join(", ", @currSyns), "\n";
foreach my \$syn (acurrSyns){
\$syn =~ /^([^\#]+)\#/; \#grab word before first '\#'
\$w = \$1;
\#if not multiword lex, add it

```
```

                    unless( $w =~ /[_ ]/ ) { #expect only _, but be safe
                        $syns{"$w:J"}=undef;
                        }
                    )
                }
            }
        }
        #add our new synlist to the adj entry of synHash
        $synHash{$adj}=[sort keys %syns];
        print "", "synonyms of $adj: ", join(", ", @{$synHash{$adj}}), "\n";
    }
    #now we have the synonyms in %syms, return them (the keys)
    return @{$synHash{$adj}};
    }
sub getAntonyms{
my ($adj) = a_;
    #unless we've already generated the synonyms on a prev iteration..
    unless( exists($antHash{\$adj}) ){
my \$word;
my \$w;
my Esenses = ();
my \&ants = ();
my @curxAnts = ();
my @currSyns = ();
\#convert into wordnet syntax
\$adj =~ /([^\:]+)\:J/;
\$word = \$1 . "\#a";
\#gen all senses
@senses = $wn->querySense($word);
\#aug3-chop sense list down to 'n' values if that option is nonzero
if( \$optsHash{'n'} ){
if( \$\#senses >= \$OptsHash{'n'} ){
splice @senses, \$optsHash{'n'}; \#chop from n onward
)
}
foreach my \$sense (@senses){
@currAnts = $wn->queryWord($sense, "ants");
\#print "'antonym' synsets:\n";
foreach my \$ant (@currAnts){
@currSyns = $wn->querySense($ant, "syns");
\#print " \$ant: ", join(", ", @currSyns), "\n";
foreach my \$syn (acurrSyns) {
\$syn =~ /^([^\#]+)\#/; \#grab the word before the first '\#'
\$w = \$1;
\#if not multiword lex, add it
unless( \$w =~ /[_ ]/ ){ \#expect only _ for multis, but be safe
$ants{"$w:J"}=undef;
}
}
)
}
\#add our new synlist to the adj entry of synHash
$antHash{$adj}=[sort keys %ants];
print "n, "antonyms of $adj: ", join(", ", @{$antHash{$adj}}), "\n";
    }
    #now we have the antonyms in %ants, return them (the keys)
    return @{$antHash{\$adj}};
}
\#\#\#END
===
\#!/home/bin/per1
\#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
\#\#\#filename: 2-log-lex-count.pl
\#\#\#author: Adam Longton, longton@cs.ubc.ca

```
\# this script is 1 of 3 that do the log-likelihood lexicon expansion.

\#description, inputs and output:
\#input: a positive lexicon, a negative lexicon, and a corpus of stemmed
\#and POS-tagged sentences, in the format
\#of the output of phase 1 script
\#the pstv and ngtv lexicons are lists of word POS pairs, one per \#line (see eg.)
\#
\#there is an option, \(-n\), that will only print lines with at least
\#one nonzero count. (this presumably reduces the data set by a large
\#fraction, down to clauses with only first generation collocations \#with lexicon words.)
\#
\#output: to standard output, each line is a clause, of form:
\#<clause-id>, <pc>, <nc> (, <stem>, <pos>) +
\#that is, a clause id, pstv lexicon-word count, ngtv lex-word-count,
\#and the stem/pos pairs for that clause.
\#
\#example:
\#
\#say you have these input files:
\#corpus.txt:
\#23, 0, 0, be:V, very:R, very:R, good:J, poor:J, nice:J
\#24, 0, 0, Amy:N, be:V, funny:J
\#
\#pos.txt:
\#excellent:J
\#good:J
\#nice:J
\#
\#neg.txt:
\#poor:J
\#bad: J
\#nasty:J
\#
\#command-line execution:
\#>2-log-lex-count.pl -n posv.txt negv.txt corpus.txt
\#
\#output:
\#23, 2, 1, be:V, very:R, very:R, good:J, poor:J, nice:J

use strict;
use warnings;
my \$usage = "usage: perl \$0 [-n] <pstvfile> <ngtvfile> <corpusfile>\n";
die Susage if (\$\#ARGV < 2);
\#\#\#process option
my \(\$ \mathrm{z}=1\); \#default is to print lines with double zero counts
my \$opts;
if (\$\#ARGV > 2) \(\{\)
\$opts = shift;
\$opts \(=\sim / \wedge-n+/\) or die \({ }^{\text {file }}\) error: \$usagen;
\$z=0; \#meaning double zero is not permitted
\}
\#process infiles
my \$pfile = shift;
my \$nfile = shift;
my \$corpfile = shift;
open(PSTV, \$pfile) or die "file error: \$usage";
open(NGTV, \$nfile) or die "file error: \$usage";
open(CORP, \$corpfile) or die "file error: \$usage";
\#hashes to hold lexicons- store items as strings in same form as
\#input, ie "stem:pos" (with the colon). this pos info is needed to \#(possibly) distinguish between homonyms of different pos. \#
\#may incorporate sense number too if WSD is done in future...
my \%pHash = ();
```

my %nHash = ();
my $line;
while ($line = <PSTV>){
chomp \$line;
$pHash{$line}=undef;
}
close(PSTV);
while (\$line = <NGTV>) {
chomp \$line;
$nHash{$line}=undef;
}
close(NGTV) ;
\#construct outlines on the fly from corpus inlines
my \$pc; my $nc;
my @lineArr;
while ($line = <CORP>) {
\$pc=0, \$nc=0; \#reset current counts
chomp $line;
    @lineArr = split /,/,$line;
for(my $i=3;$i<=$#lineArr;$i++){
$pc++ if( exists($pHash{$lineArr[$i]}) );
$nc++ if( exists($nHash{$lineArr[$i]}));
}
$lineArr[1]=$pc; \#update counts
$1ineArr[2]=$nc;
if(\$z || \$pc>0 || \$nc>0){ \#consider non-zero option
print join(',',@lineArr);
print "\n";
}
}
close(CORP);
\#\#\#END
===
\#!/home/bin/perl
\#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
\#\#\#filename: 3-1og-word-stats.pl
\#\#\#author: Adam Longton, longton@cs.ubc.ca

# this script is 2 of 3 that do the log-likelihood lexicon expansion.

\#*****************************************************************
\#description, inputs and output:
\#input: the output of phase 2 (see its comments). it is a file
\#of clauses with counts of pstv and ngtv lexicon word occurences.

# 

\#the options are just like phase l - they allow you to filter for
\#specific parts of speech. this is a separate operation from the
\#filtering in phase 1. here it is so you can generate a stats file
\#on only those pos's you're interested in (presumably the ones that you
\#predict will carry polarity).

# 

\#output: to standard output each line is stats of a word, of form:
\#<stem>:<pos>,<ic>,<pic>,<nic>,<pcc>,<ncc>
\#where
\#ic = instance count, the number of occurences of that word in pstv
\#or ngtv contexts. a pstv (ngtv) 'context' is a clause that
\#has at least one pstv (ngtv) lexicon word in it.
\#pic = pstv instance count, number of occurences of that word in pstv
\#contexts
\#nic = same as pic, but for ngtv contexts
\#pcc = pstv collocation count, the number of pstv lexicon words
\#that co-occur in clauses with the word
\#ncc = same as pcc, but for ngtv lex words

# 

\#note - for the special case of the same word appearing multiple
\#times in a clause, each instance is processed separately, so
\#instances and collocations all get multiply counted in these cases

```
```

\#(even for the same instances of lexicon words).

# 

\#the output will be in alphabetical order.

# 

\#the first four lines are totals for the 4 parts of speech

# 

\#example:

# 

\#say you have this input file:
\#corpus.txt:
\#23,2,1,be:V, very:R,very:R, good:J, poor:J, nice:J
\#24,1,0,Amy:N,be:V, exceptionally:R, funny:J,nice:J

# 

\#command-line execution:
\#>3-log-word-stats.pl corpus.txt

# 

\#output:
\#N,1,1,0,1,0
\#V,2,2,1,3,1
\#丁,5,5,3,8,3
\#R,3,3,2,5,2
\#Amy:N,1,1,0,1,0
\#be:V,2,2,1,3,1
\#exceptionally:R,1,1,0,1,0
\#funny:J,1,1,0,1,0
\#good:J,1,1,1,2,1
\#nice:J,2,2,1,3,1
\#poor:J,1,1,1,2,1
\#very:R,2,2,2,4,2
\#*****************************************************************
use strict;
use warnings;
\#\#() for now don't impl onlyfile
my \$usage = "usage: perl \$0 -nvjr <corpusfile>\n";
die $usage if ($\#ARGV < 0);
my %optsHash = ('n',1,'v',1,'j',1,'r',1); \#flags, default all on
\#\#\#process options
my $opts;
my goptsArr;
if($\#ARGV > 0){
\$opts = shift;
\$opts =~ /^-/ or die "file error: $usage";
    if($opts eq '-'){die "file error: \$usage";}
$opts = substr($opts,1);
if(\$opts =~ /[^nvjr]/){die "file error: $usage";}
    @optsArr = split //.$opts;
%optsHash = ('n',0,'v',0,'j',0,'r',0); \#reset
foreach my \$c (@optsArr){
$optsHash{$c}=1;
}
}
\#process infiles
my \$line;
\#process corpus input file
my \$corpfile = shift;
open(CORP, \$corpfile) or die "file error: \$usage";
\#word hash - keys are stem:pos strings (just like input) and
\#values are refs to anon arrays of
\#size 5, for ic,pic,nic,pcc,ncc counts
my %wHash = ();
\#totals hash
my %tHash = ('N' => [0,0,0,0,0],
'V'=> [0,0,0,0,0],
'J' => [0,0,0,0,0],

```
\[
\left.'^{\prime} R^{\prime}=>(0,0,0,0,0]\right) ;
\]
\#generate and update word hash entries from corpus inlines \#keep running global totals counts..
```

my \$pc; my \$nc;
my @lineArr;
my $pos;
while ($line = <CORP>) {
chomp $line;
    @lineArr = split /,/,$line;
$pc=$lineArr[1];
$nc=$lineArr[2];
for(my $i=3;$i<=$#lineArr;$i++){
\$pos = substr($1ineArr[$i],-1,1);
\#\#added may30--need pos-filtering here, not in step 1
if( \$pos eq 'N' \&\& \$optsHash{'n'} || \$pos eq 'V' \&\& \$optsHash{'v'} || \$pos eq 'J'
\&\& \$optsHash{'j'} || \$pos eq 'R' \&\& $optsHash{'r'} ){
            unless( exists($wHash{$1ineArr[$i]}) ) {
\#initialize the anon count array for curr word
\$wHash{$1ineArr[$i]}=[0,0,0,0,0];
}
\$wHash{$1ineArr[$i]}->[0]++;
\$wHash{$1ineArr[$i]}->[l]++ if \$pc;
\$wHash{$1ineArr[$i]}->[2]++ if \$nc;
$wHash{$lineArr[\$i]}->[3] += \$pc;
$wHash{$lineArr[\$i]}->[4] += $nc;
                #update running totals
                #$pos = substr($lineArr[$i],-1,1);
$tHash{$pos}-> [0]++;
$tHash{$pos} -> [1]++ if \$pc;
$tHash{$pos}->[2]++ if \$nc;
$tHash{$pos}->[3] += \$pc;
$tHash{$pos} -> [4] += \$nc;
}
}
}
close(CORP);

```
\#print totals as first 4 lines of output
foreach my \$pos ('N', 'V','J','R')\{
    print \$pos . ',';
    print join(',', @\{\$tHash\{\$pos\}\});
    print " \(\backslash \mathrm{n}\) ";
\}
\#print the words with their counts, in lexical order
foreach my \$word (sort keys \%wHash) \{
    print \$word . ',';
    print join(',', @\{\$wHash\{\$word\}\});
    print " \(\backslash n\) ";
\}

\section*{\#\#\#END}
===
\#!/home/bin/perl
\#123456789 123456789123456789123456789123456789123456789123456789123456789
\#\#\#filename: 4-1og-word-scores.p1
\#\#\#author: Adam Longton, longton@cs.ubc.ca
\# this script is 2 of 3 that do the log-likelihood lexicon expansion.

\#description, inputs and output:
\#input: the output of phase 3 (see its comments). it is a file
\#of lines of 6 comma separated fields. they are stem:pos entries with
```

\#5 various polarity word collocation counts.

# 

\#output: word, score pairs (score defnd by [YuHatz03] eqn)

# 

\#example:

# 

\#input file, word-stats.txt:
\#N,1,1,0,1,0
\#V,2,2,1,3,1
\#J, 5, 5, 3, 8, 3
\#R,3,3,2,5,2
\#Amy:N,1,1,0,1,0
\#be:V,2,2,1,3,1
\#exceptionally:R,1,1,0,1,0
\#funny:J,1,1,0,1,0
\#good:J,1,1,1,2,1
\#nice:J,2,2,1,3,1
\#poor:J,1,1,1,2,1
\#very:R,2,2,2,4,2

# 

\#command-1ine execution:
\#>4-log-word-scores.pl word-stats.txt

# 

\#output:
\#Amy:N, 0
\#be:V,0
\#exceptionally:R,0.310154928303839

# funny:J,0.211309093667207

\#good:J,-0.376477571234912
\#nice:J,-0.040005334613699
\#poor:J,-0.376477571234912
\#very:R,-0.200670695462151

# 

\#*****************************************************************
use strict;
use warnings;
my \$usage = "usage: perl \$0 <word-stats-file>\n";
die $usage if ($\#ARGV < 0);
\#define epsilon for smoothing (for zero counts)
my \$ep=0.5;
\#process infile
my \$wordfile = shift;
open(WFILE, \$wordfile) or die "file error: \$usage";
my \$word; my \$ic; my \$pic; my \$nic; my \$pcc; my \$ncc;
my \$score;
my \$1ine;
my \$pos;
my @tempArr;
\#grab totals info from 1st 4 1ines..
my %tHash = ('N'=>[],'V'=>[],'J'=>[],'R'=>[]);
for(my $i=0;$i<4;\$i++) {
\$1ine = <WFILE>;
chomp $line;
    OtempArr = split /./,$line;
\$pos = shift @tempArr;
for(my $j=0;$j<5;$j++){
        push( @{$tHash{\$pos}}, $tempArr[$j]);
}
}
\#precalc global score offsets for each pos. tack it onto the
\#anon array in tHash
\#note that the offset is +totalncc-totalpcc
my \$os;
my \$tncc; my \$tpcc;

```
```

foreach my \$pos ('N','V','J','R'){
$tncc=$tHash{\$pos}-> [4];
$tpcc=$tHash{$pos}-> [3];
    #these cases shouldn't occur in a full corpus, only here for testing
    #$tncc = 0.5 if $tncc == 0;
    #$tpcc = 0.5 if \$tpcc == 0;
$os = log($tncc + $ep) - log($tpcc + $ep);
    #$os = 0; \#\#\#tmp
push @{$tHash{$pos}}, \$os;
\#print \$pos. ',' . $os . ',' . exp($os) . "\n";
}
my $expscore;
#print the words with their scores, in lexical order
while ($line = <WFILE>) {
chomp $line;
    ($word,$ic,$pic,$nic,$pcc,$ncc) = split /,/,$line;
$pos = substr($word,-1,1); \#last char
$score = log($pcc + $ep) - log($ncc + \$ep) + $tHash{$pos}-> [5];
$expscore = sprintf( "%.5f", exp($score) );
print "$word,$score, $expscore,$ic,$pic,$nic,$pcc,$ncc\n";
}
close(WFILE);
\#\#\#END
===
\#!/home/bin/perl
\#123456789 123456789 123456789 123456789 1234567.89 123456789 123456789 123456789
\#\#\#filename: 5-score-corp.p1
\#\#\#author: Adam Longton, longton@cs.ubc.ca

# this script takes a scored lexicon and a formatted tagged annotated corpus and predicts

the polarity of the sentences

# in the corpus.

# It has options j n v r p m c d g e and b.

# j n v r are the parts of speech you want to score in the corpus.

# p - "presupposition" context shifter

# m - "modal" context shifter

# c - "contrastive" context shifter

# d - "deflag" option. these are "stop" words that signal a break in the negation context

window. basically it

# provides a coarse estimate of syntactic boundaries, as far as negation words are

concerned. if a stop word

# comes after a negation word, it "deflags" this negation word, reducing its effect to

nothing.

# g - "negation" context shifter

# e - effective opinion processing (tie breaking and optionally but-logic)

# b - "but" context shifter or effective opinion but-logic (toggled in code)

\#*****************************************************************
use strict;
use warnings;
my \$usage = "usage: perl \$0 -jnrvdgbcmpxe <scorefile> <corpusfile> [<negWindow>]
[g<addNegWord>] [c<addConWord>] [def<defaultScoreCode>]\n";
my \$minArgs = 3; \# opts letters after the "-" are manditory. this simplifies the optional
params like negWindow.
my \$minLastIndexOfArgv = \$minArgs - 1;
die $usage if ($\#ARGV < \$minLastIndexOfArgv);
my %optsHash = ('n',1,'v',1,'j',1,'r',1,'g',1,'b',1,'`',1,'c',1,'m',1,'p',1,'x',1,'e',1);
\#process options
my \$optsstr = "n;
my \$opts;
my goptsArr;
\$opts = shift;
\$optsStr = \$opts;
\$opts =~ /^-/ or die "file error: \$usage";

```
```

if(\$opts eq '-'){die "file error: \$usage";}
$opts = substr($opts,1);
if(\$opts =~ /[^nvjrgbdcmpxe]/){die "file error: $usage";}
QoptsArr = split //,$opts;
%optsHash = ('n',0,'v',0,'j',0,'r',0,'g',0,'b',0,'d',0,'c',0,'m',0,'p',0,'x',0,'e',0);
\#reset
foreach my \$ch (@optsArr){
$optsHash{$ch}=1;
}
\#get input files and params
\#throw word scores file into a hash (word/score as key/value)
my \$scorefile = shift;
open(SCOR, \$scorefile) or die "file error: \$usage";
my %wHash = ();
my @lineArr;
my GannArr;
my $line;
while ($line = <SCOR>) {
chomp $line;
    alineArr = split /,/,$line;
\#only add cand scoredwords (for nvjr subsets)
if( \&isCandidateString(\$lineArr[0]) ) {
$wHash{$lineArr[0]}=\$lineArr[l];
}
}
close(SCOR);

# process other params:

my \$corpfile = shift;
open(CORP, \$Corpfile) or die "file error: \$usage";
\#append the outputfile with the same suffix found in the input corpus file
my \$corpTag = n";
if( \$corpfile =~ /lout(-.+)\.txt/ ){
\$corpTag = \$1;
}
\#some params of the corp scoring loop further down. moved up to here for optional
overrides.
my \$negWindow = 6; \#5 is better for j only
my \$butWindow = 9999; \#for advanced but as negation, inf window (ie to beginning of
sentence...)
my \$conWindow = 8;

# optional overrides

my \$negWinstr = n";
my $negWinArg = " ";
if($\#ARGV >= 0){ \#means we have further args we haven't shifted out yet
\$negWindow = shift;
$negWindow =~ /^\d+$/ or die "file error: \$usage"; \#must be an posv integer
\$negWinArg = \$negWindow;
$negWinStr = "-$negWindow";
}
my \$addNegWordStr = "n;
my $addNegWordArg = "n;
if($\#ARGV >=0){ \#optionally add "no" to neg list. hurts adj only expts but helps jnrv.
use 0/1 as bools and req prepend with g.
\$addNegWordArg = shift;
$addNegWordArg =~ /^g[01]$/ or die "file error: \$usage, bad arg: \$addNegWordArg";
$addNegWordStr = "-$addNegWordArg";
}
my \$addConWordStr = "n;
my $addConWordArg = n";
if($\#ARGV >=0) { \#optionally add "while" to contrastives list. use 0/l as bools and req
prepend with c.
\$addConWordArg = shift;
$addConWordArg =~ /^c[0l]$/ or die "file error: \$usage, bad arg: \$addConWordArg";
$addConWordStr = "-$addConWordArg";
}

```
```

my \$defScoreStr = N";
my $defscoreArg = "N;
if($\#ARGV >=0) { \#optionally specify the default scoring algortihm for ties and nulls. 0-
alwaysNegv, I-alwaysPosv, 2-UsePrevSentScore,3-alternate+and-. req prepend with def.
\$defScoreArg = shift;
$defScoreArg =~ /^def[0123]$/ or die "file error: \$usage, bad arg: \$defScoreArg";
$defScorestr = "-$defScoreArg";
}
\#set up output files
open(RES,
">results\${ corpTag} ${optsStr}${negWinStr} ${addNegWordStr}${addConWordStr}\${defScoreStr}.t
xt") or die "error trying to open file1 for writing";
open(ACC, ">acc.txt") or die "error trying to open file2 for writing";
\#reconstruct arguments of this script call and record this in the results output file for
reference
my \$commandAndArgs = "perl \$0 \$optsStr \$scorefile \$corpfile \$negWinArg \$addNegWordArg
\$addConWordArg \$defScoreArg";
print RES "command: \$commandAndArgs\n";
\#***********************************************************************
\#now score each test line by averaging its words that are
\#in the word score hash. default score is zero (neutral polarity).
my \$score;
my \$n; \#number of scored words found in the current sentence
\#counters
my \$nSuccNonZero = 0; my \$nSuccZero = 0; my \$nSuccNull = 0;
my \$nNonZero = 0; my \$nzero = 0; my \$nNull = 0;
my \$nSuccNonZeroEff = 0; my \$nNonZeroEff = 0;
my \$nSuccCandNull = 0; my \$nCandNull = 0;
my \$candFlag = 0;
my \$cands = " ";
\#internal flag-switch for using nulls or counting null words as zero.
my \$useZeros = 0;
\#needed for formatted number output
my \$number = 0; my \$rounded = 0;
\#previous binary score holder for default scoring in 0.000 and null cases
my \$prevScore = '+'; \#note the arbitrariness of + vs -

```
```

\#neg and but flags

```
#neg and but flags
my $negX; #negation multiplier l or -1
my $negX; #negation multiplier l or -1
my $negCount;
my $negCount;
my $butX; my $butCount; my $butFlag = 0;
my $butX; my $butCount; my $butFlag = 0;
my $conX; my $conCount; my $conFlag = 0;
my $conX; my $conCount; my $conFlag = 0;
my %deflagHash = ();
my %deflagHash = ();
if( $optsHash{'d'} ) {
if( $optsHash{'d'} ) {
    %deflagHash =
    %deflagHash =
("but",0,"except",0,"however",0,"only",0,"although",0,"though",0,"while",0,"whereas",0,"-
("but",0,"except",0,"however",0,"only",0,"although",0,"though",0,"while",0,"whereas",0,"-
",0,"SC",0);
",0,"SC",0);
}
}
my %contrastHash = ();
my %contrastHash = ();
if( $optsHash{'c'} ){
if( $optsHash{'c'} ){
    8contrastHash = ("although",0,"despite",0, "while",0);
    8contrastHash = ("although",0,"despite",0, "while",0);
    if($addConWordArg eq "cO"){
    if($addConWordArg eq "cO"){
        %contrastHash = ("although",0,"despite",0);
        %contrastHash = ("although",0,"despite",0);
    }
    }
}
}
my %modalHash = ();
my %modalHash = ();
if( $optsHash{'m'} ){
if( $optsHash{'m'} ){
    %modalHash = ("would",0, "should",0);
    %modalHash = ("would",0, "should",0);
    #%modalHash = ("would",0,"should",0,"could",0);
    #%modalHash = ("would",0,"should",0,"could",0);
}
}
my %presupHash = ();
my %presupHash = ();
if( $optsHash{'p'} ){
if( $optsHash{'p'} ){
    %presupHash =
    %presupHash =
("miss",0," forget",0, "refused",0,"assumed",0,"hard",0, "harder",0, "less",0);
("miss",0," forget",0, "refused",0,"assumed",0,"hard",0, "harder",0, "less",0);
}
```

}

```
```


# to be investigated:

my %excessHash = ();
if( \$optsHash{'x'} ){
%excessHash = ("overly",0);
\#%excessHash = ("overly",0,"too",0,"enough",0);
}
my %negHash = ();
if( $optsHash{'g'} ){
    %negHash =
("not",0,"never",0,"n't",0, "doesnt",0,"'t",0, "cannot",0, "nothing", 0, "nor",0, "dont",0, "wou
ldnt",0,"no",0);
    if($addNegWordArg eq "g0"){
%negHash =
("not",0,"never",0,"n't",0,"doesnt",0,"'t",0, "cannot", 0, "nothing", 0, "nor",0,"dont",0,"wou
ldnt",0); \#redefine w/o no.
}
}
my %butHash = ();
if( \$optsHash{'b'} ){
%butHash = ("but",0, "except",0,"however",0);
\#to be investigated: yet though whereas despite nevertheless nonetheless
}
my \$negDistance;
my \$butDistance;
my \$conDistance;

# but variables

my \$advancedBut = 1; \#enables "but" shifter for but-as-negation
my \$butAsNot = 1; \# 0 means just exclude don't invert affected words
my \$butDefaultFlip = 0; \# experimental
my \$effButEnabled = 0; \# used wih effective opinion to do hu-liu but-conjunction logic.
if this is l \$advancedBut must be 0 and vice versa
my @wordArr;
my \$advCon = 1;
my \$conAsNot = 1;
\#for the flip def score alg, for corps of isolated setences
my \$zflip = '-';
my \$nflip = '-';
\#------------------------------------------------------------------------------

# loop through the corp file and score each line

while (\$line = <CORP>){
chomp $line;
    @lineArr = split /,/,$line;
\$score = 0;
\$n = 0;
my \$numEffOpinions = 0; \#will be the number of found effective opinions that
contributed to the score, for the sake of averaging later.
my @predArr = (0,0,0,0); \#fourth elem for eff tag, moved inside the while loop
\$negx = 1;
\$negCount = -1;
\$butX = 1;
\$butCount = -l;
\$butFlag = 0; \#0 no 1 yes
\$conX = 1;
\$conCount = -1;
\$conFlag = 0;
my \$featureStr = \$lineArr[2];
\$candFlag = 0;
$cands = &isCandidateString($line);
if( \$cands ){
\$candFlag = 1;
\$lineArr[2] = $lineArr[2] . ";CANDS\=$cands"; \#feature list support
}
else{

```
```

    $lineArr[2] = $lineArr[2] . ";0";
    }
\#***********************compute predicted score
\#first left to right thru sent, for neg processing..
\#this sec also used for most tagging (post-neg shifters)
for(my $i=3;$i<=$#lineArr;$i++){
\#context shifter processing:
\#negcount
if( \$negCount >= 0 ){ \#careful with >= and <
\$negCount--;
if( \$negCount < 0 ){
$negX = 1;
    }
    #deflagging
    elsif( &isDeflagWord($lineArr[\$i]) ){
$negX = 1;
    }
}
#check for negation words, if so, flip neg flag
if( &isNegWord($lineArr[\$i]) ){
\$negX = -1;
\$negCount = \$negWindow; \#\#reset neg window
$lineArr[$i] = $lineArr[$i] . ";" . "NEG:$negWindow";
}
#flag but words (no window for default)
if( &isButWord($lineArr[\$i]) ) {
if( \$i == 3 ){ \#only set flag for def if at beg of sent
\$butFlag = l;
$lineArr[$i] = $lineArr[$i] . ";" . "BUTS"; \#special tag
}
else{
\$butFlag = 1; \#exluding helps Al1, hurts J
$lineArr[$i] = $lineArr[$i] . ";" . "BUT";
}
}
\#contrastives
if( \&isContrastWord($1ineArr[$i]) ) {
$conFlag = 1;
    if($i == 3){
$lineArr[$i] = $lineArr[$i] . ";" . "CONS"; \#special tag
}
else{
$lineArr[$i] = $lineArr[$i] . n;" . "CON";
}
}
\#tag modals
if( \&isModalWord($lineArr[$i]) ){
\#grab rest of sent, try to pattern match..
my $rest = join(' ',@lineArr[$i+1 .. \$\#lineArr]);
\#note pattern accounts for stuff like: would not have ever been..
\#might want other-than R's, say 'sort:? of:?'
if( \$rest =~ /^(\S+\:R )*be\:/ ||
\$rest =~ /^(\S+\:R )*have\:\S+ (\S+\:R )*been\:/ ){
\$conFlag = 1; \#just combine processing with cons for inverting shifters
$lineArr[$i] = $lineArr[$i] . ";" . "MOD";
}
}
if( \&isPresupWord($lineArr[$i]) ) {
if( $lineArr[$i] =~ /^hard/ ) { \#hard, harder
if( \$i < \$\#lineArr ){
if( $lineArr[$i+l] =~ /^to\:/ ) {
\$conFlag = 1;
$lineArr[$i] = $1ineArr[$i] . ";" . "PRE";
}
}
}
else{
\$conFlag = 1;
$lineArr[$i] = $lineArr[$i] . ";" . "PRE";

```
```

    }
    }
    if( &isExcessWord($lineArr[$i]) ){
    $conFlag = 1;
    $lineArr[$i] = $lineArr[$i] . ";" . "EXC";
    }
    #only try to score candidate words (ie, of valid POS)
    if( &isCandidateString($lineArr[$i]) ) {
    if( exists($wHash{$lineArr[$i]}) ) {
        $number = $negX * $wHash{$lineArr[$i]}; #neg
        $score = $score + $number;
        $n++;
        $rounded = sprintf("%.3f", $number);
        $lineArr[$i] = $lineArr[$i] . ";" . $rounded; #if want individual word scores
    appended
if( \$negx == -1 ){
\$negDistance = \$negWindow - \$negCount;
$lineArr[$i] = $lineArr[$i] . ";" . "NGD:\$negDistance";
}
}
}
}
\#\#contrastives and other inverting shifters
if( \$advCon \&\& \$conFlag ){
for(my $i=3;$i<=$#lineArr;$i++) {
if( \$conCount >= 0 ){ \#careful with >= and <
\$conCount--;
if( \$conCount < 0 ){
$conX = 1;
            }
            elsif( &isDeflagWord($lineArr[\$i]) ) {
\$conX = 1;
}
}
\#mod
if( $lineArr[$i] =~ /CON/ ||
$lineArr[$i] =~ /MOD/ ||
$lineArr[$i] =~ /PRE/ ){
\$conX *= -1;
\$conCount = \$conWindow;
$lineArr[$i] = $lineArr[$i] . ":\$conWindow";
}
elsif( $lineArr[$i] =~ /\;-?\d/ ){ \#is a scored word
if( \$conX == -1 ){
\$conDistance = \$conWindow - \$conCount;
$lineArr[$i] = $lineArr[$i] . ";" . "CND:$conDistance";
                        #then flip sign of word score,adjust sentence score:
                    @wordArr = split /;/,$lineArr[\$i];
\$wordArr[1] = (-1) * \$wordArr[1];
\$score += \$wordArr[1]; \#nullifies original contribution
if( \$conAsNot ) {
\$score += \$wordArr[1];
} \#othe rdirection
else{
\$wordArr[l] = 0; \#annotate it as 0.000 (nullified)
}
\$wordArr[1] = sprintf("%.3fn, \$wordArr[1]);
$1ineArr[$i] = join(';',(0wordArr);
}
}
}
}

```
\#now proc right to left for advanced but-as-not scoring... comes after the cons section cuz it has weaker precedence.
if ( SadvancedBut \&\& \$butFlag ) \{
for (my \$i=\$\#lineArr; \$i>=3; \$i--) \{
if( \$butcount >= 0)\{ \#careful with >= and <
\$butCount--;
```

            if( $butCount < 0 ){
                        $butx = 1;
                        }
        }
        if( $lineArr[$i] =~ /BUT/ ){
            $butX = -1;
            $butCount = $butWindow;
            $lineArr[$i] = $lineArr[$i] . ":$butWindow";
        }
            elsif( $lineArr[$i] =~ 八\-?\d/ ){ #is a scored word
            if( $butx == -1 ) {
                $butDistance = $butWindow - $butCount;
                $lineArr[$i] = $lineArr[$i] . ";" . "BTD:$butDistance";
                    #then flip sign of word score,adjust sentence score:
                    @wordArr = split /;/,$lineArr[$i];
                    $wordArr[1] = (-1) * $wordArr[1];
                    $score += $wordArr[1]; #nullifies original contribution
                    if( $butAsNot ){
                    $score += $wordArr[1];
                } #otherdirection
                else{
                    $wordArr[1] = 0; #annotate it as 0.000 (nullified)
                    }
                    $wordArr[l] = sprintf("%.3f", $wordArr[1]);
                $lineArr[$i] = join(';',@wordArr);
            }
            }
        }
    }
    #******************* apply effective-opinion tie breaking and but-conjunction logic if
    requested
if( \$optsHash{'e'} \&\& $n > 0 && ($score == 0 || (\$effButEnabled \&\& \$butFlag)) ){
\#for each feature word, try to find it in the sentence. if found, find the closest
scored word.
\#add that score to the current score.
if( \$featureStr =~ /FEATURES=(.+)/ ){
my \$effScore = 0;
\#if this line contains a but word, consider the line to begin after the first
but word.
my $firstButIndex = 3;
            if($butFlag) { \#find the index of the word after the first but word
for(my \$m = 3; \$m <= \$\#lineArr; $m++){
            if( &isButWord($lineArr[\$m]) ){
\$firstButIndex = \$m;
last;
}
}
}
my \$fStr = $1;
        my @featureArr = split /:/,$fStr;
L1: foreach my \$feature ( @featureArr ) {

            $feature =~ s/^([^\[]+)\[.*/$1/; #throw away the trailing score stuff
            $feature =~ s/^(.+)_$/$l/; #just in case a feature ended with a "_"
            my @currFeatArr = split /_/,$feature;
            # ^general case n-word feature, simple case is 1-word (ie this arr is length
    1)          my $firstWordIndex = 3; # 3 is the index of the first word in the sentence
         if($butFlag){
                         $firstWordIndex = $firstButIndex + 1; #start after the first but word
                 }
                 #try to find index in sentence of $currFeatArr[0]
                 my $i = $firstWordIndex;
                 L2: for( ; $i <= $#lineArr; $i++ ){
                 $lineArr[$i] =~ /^([^:]+):/; #grab word
                 my $word = $1;
                 if( &stemMatch( $currFeatArr[0], $word ) ){
                     last L2 if( $#currFeatArr == 0 ); #1-word feature found
                 #if number words left in the feature > number words left in sentence:
                 if( $#CurrFeatArr > ($#1ineArr - $i) ){
                 $i = $#lineArr + 1; #fall off the end of the sentence
    ```
```

                last L2;
            } #else attempt to match the whole phrase:
            L3: for( my $j = 1; $j <= $#currFeatArr; $j++ )(
                    $lineArr[$i+$j] =~ /^([^:]+):/; #grab word
                    $word = $1;
                    next L2 unless( &stemMatch( $currFeatArr[$j], $word ) );
                    }
                            last L2; #n-word feature found
            }
        }
        next L1 if( $i > $#lineArr ); #feature not found
        #otherwise find its effective opinion (ie the score of the nearest scored-
    word). search left and right,
\#tracking score of min distance scored-word. search left:
my \$leftDist = 0;
my \$leftEffOpinion = 0;
for( my \$j = \$i - 1; \$j >= \$firstWordIndex; \$j-- ){
grab score
\$leftEffopinion = \$1;
\$leftDist = \$i - \$j;
last;
}
}
my \$rightDist = 0;
my \$rightEffOpinion = 0;
for( my \$j = \$i + \$\#currFeatArr + 1; \$j <= \$\#lineArr; \$j++ ){
if( $lineArr[$j] =~ /^[^:]+:[^:]+;(-?[\d]+\.[\d]+)/ )( \#if word is scored,
grab score
\$rightEffOpinion = \$1;
\$rightDist = $j - ($i + \$\#currFeatArr);
last;
}
}
unless( \$leftDist == 0 \&\& \$rightDist == 0 ){ \#if an effective opinion was
found
my \$effopinion = 0;
if( \$rightDist > 0 \&\& ( \$leftDist == 0 || \$rightDist <= \$leftDist ) ) {
\$effOpinion = \$rightEffOpinion;
}
else(
\$effOpinion = \$leftEffOpinion;
}
\$effScore += \$effOpinion;
my \$rounded = sprintf("%.3f", \$effopinion);
\#we found an eff opinion and added it to our sentence score. mark the
feature with it:
for( my \$j = \$i; \$j - \$i <= \$\#currFeatArr; $j++ ){
            #should be bare (no ";"), but just in case it was also scored or a
marked shifter:
            my @wArr = split /;/,$lineArr[\$j];
my \$marker = "EFD";
if( \$j - \$i == \$\#currFeatArr ) {
\$marker = $marker . ":$rounded";
\$numEffOpinions++;
}
push @wArr, \$marker;
$lineArr[$j] = join ";",@wArr;
}
}
}
\# update the line score with the effScore
if( $effScore != 0 ){
        if($butFlag) (
$predArr[3] = "EFB:$numEffopinions"; \#mark the sentence if the score was
determined by the effective-opinion-but rule
}
else{
$predArr[3] = "EFF:$numEffopinions"; \#mark the sentence if the score has
become nonzero via effective opinion tie breaking

```
```

    }
    $score = $effScore;
    }
    elsif($butFlag && $score != 0) {
        $score *= -1; #if no eff opinion found past the but word, we invert the
    originally found score
$predArr[3] = "EFN:$numEffOpinions"; \#mark the sentence if the score has was
flipped because of an effective-opinion-free but-clause
}
}
}
\#***********************set score, outcome and counts
\#\#grab annarr
@annArr = split /;/,$lineArr[0];
    #********null case
    if($n == 0){
\$predArr[2] = "null";
\$predArr[1] = \&nullDefaultScore();
\#cmp to ann
if( SannArr[1] eq '+' ){
if( \$predArr[1] eq '+' ) {
\$predArr[0] = "SuCCESS";
\$nSuccNu:11++;
if( $candFlag ){$nSuccCandNull++;}
}
else{
\$predArr[0] = "FAIL";
}
\$nNull++;
if( $candFlag ){$nCandNul1++; }
}
elsif( \$annArr[1] eq '_' ){
if( \$predArr[1] eq '-' ){
\$predArr[0] = "SUCCESS";
\$nSuccNull++;
if( $candFlag ) {$nSuccCandNull++;}
}
else{
\$predArr[0] = "FAIL";
}
\$nNull++;
if( $candFlag ) {$nCandNul1++;}
}
else{ \#won't happen unless have m and 0 anns
\$predArr[0] = "IGNORE";
}
}
else{ \#*************nonnull case, ie n>0
if( \$numeffopinions > 0 ){
\$score = $score/$numEffOpinions;
}
else{
\$score = $score/$n; \# average, but could also use raw score
}
\$rounded = sprintf("%.3f", \$score);
\#set float score
\$predArr[2] = \$rounded;
\#********zero case - tie
if( \$predArr[2] == 0 ){
\$predArr[1] = \&zeroDefaultScore();
\#cmp to ann
if( \$annArr[1] eq '+' ){
if( \$predArr[1] eq '+' ) {
\$predArr[0] = "SUCCESS";
\$nSuccZero++;
}
else{

```
```

                $predArr[0] = "FAIL";
            }
            $nZero++;
        }
        elsif( $annArr[1] eq '-' ) {
            if( $predArr[1] eq '-' ){
                $predArr[0] = "SUCCESS";
                $nSuccZero++;
            }
            else{
                $predArr[0] = "FAIL";
            }
            $nZero++;
    }
    else{ #won't happen unless have m and 0 anns
        $predArr[0] = "IGNORE";
    }
    }
\#**********nonzero case
else{
\#+ or -?
if( \$predArr[2] > 0 ){
\$predArr[1] = '+';
}
elsif( \$predArr[2] < 0 ){
\$predArr[1] = '-';
}
else{ \#unexpected
print STDERR "ERROR, sent score was \$predArr[2]\n";
}
\#cmp to ann
if( \$annArr[1] eq '+' ) {
if( \$predArr[1] eq '+' ){
\$predArr[0] = "SUCCESS";
\$nSuccNonZero++;
if( \$numEffOpinions > 0 ){
\$nSuccNonZeroEff++;
}
}
else{
\$predArr[0] = "FAIL";
}
\$nNonZero++;
if( \$numEffOpinions > 0 ){
\$nNonZeroEff++;
}
}
elsif( \$annArr[1] eq '-' ) {
if( \$predArr[1] eq '_' ){
\$predArr[0] = "SUCCESS";
\$nSuccNonZero++;
if( \$numEffOpinions > 0 ){
\$nSuccNonZeroEff++;
}
}
else{
\$predArr[0] = "FAIL";
}
\$nNonZero++;
if( {numEffOpinions > 0 ){
\$nNonZeroEff++;
}
}
else{ \#won't happen unless have m and 0 anns
\$predArr[0] = "IGNORE";
}
}
}
\#record current predicted binary score as prevScore
\$prevScore = \$predArr[1];

```
```

    #put predArr into lineArr[l] (lineArr[2] has FEAT and CAND info)
    $1ineArr[l] = join(';',OpredArr);
    #reassemble line and print
    print join(',',0lineArr);
    print "\n\n";
    }
close(CORP)
\#**************************************************************************
\#now stats info at end
\#composite counts
my \$nSuccTotal = \$nSuccNonZero + \$nSuccZero + \$nSuccNull;
my \$nTotal = \$nNonZero + \$nZero + \$nNu11;
my \$nSuccNonNul1 = \$nSuccNonZero + \$nSuccZero;
my \$nNonNull = \$nNonZero + \$nZero;
my \$nSuccCand = \$nSuccNonZero + \$nSuccZero + \$nSuccCandNull;
my \$nCand = \$nNonZero + \$nZero + \$nCandNul1;
my \$nSuccNonZeroReal = \$nSuccNonZero - \$nSuccNonZeroEff;
my \$nNonZeroReal = \$nNonZero - \$nNonZeroEff;
\#accuracies
my \$accTotal = 0; my \$accNonNull = 0; my \$accNonZero = 0; my \$accZero = 0;
my $accNull = 0;
if($nTotal != 0){\$accTotal = sprintf("%.5f", $nSuccTotal/$nTotal); }
if($nNonNul1 != 0) {$accNonNull = sprintf("%.5f", $nSuccNonNull/$nNonNull);}
if($nNonZero != 0){$accNonZero = sprintf("%.5f", $nSuccNonZero/$nNonZero);}
if($nZero != 0) {$accZero = sprintf("%.5f", $nSuccZero/$nZero);}
if($nNull != 0){$accNull = sprintf("%.5f", $nSuccNull/$nNull);}
my \$accCand = 0; my $accCandNull = 0;
if($nCand != 0) {\$accCand = sprintf("%.5f", $nSuccCand/$nCand);)
if($nCandNull != 0) {$accCandNull = sprintf("%.5f", $nSuccCandNull/$nCandNull);}
my \$accNonZeroReal = 0; my $accNonZeroEff = 0;
if($nNonZeroReal != 0){\$accNonZeroReal = sprintf("%.5f",
$nSuccNonZeroReal/$nNonZeroReal);}
if($nNonZeroEff != 0){$accNonZeroEff = sprintf("%.5f", $nSuccNonZeroEff/$nNonZeroEff);}
\#compute coverages
my \$cvgNonNull = 0; my \$cvgNonZero = 0; my $cvgCand = 0;
if($nTotal != 0){
\$cvgNonNull = sprintf("%.5f", $nNonNull/$nTotal);
\$cvgNonZero = sprintf("%.5f", $nNonZero/$nTotal);
\$cvgCand = sprintf("%.5f", $ncand/$nTotal);
}
print "nSuccTotal/nTotal \t= $nSuccTotal/$nTotal \t= \$accTotal\n";
print "nSuccCand/nCand \t= $nSuccCand/$nCand \t= \$accCand\n";
print "nSuccNonNull/nNonNull \t= $nSuccNonNull/$nNonNull \t= \$accNonNull\n";
print "nSuccNonZero/nNonZero \t= $nSuccNonZero/$nNonZero \t= \$accNonZero\n";
print "nSuccZero/nZero \t= $nSuccZero/$nZero \t= \$accZero\n";
print "nSuccNull/nNull \t= $nSuccNull/$nNull \t= \$accNull\n";
print "nSuccCandNull/nCandNu11 = $nSuccCandNull/$nCandNul1 \t= \$accCandNull\n";
if( \$optsHash{'e'} ){
print "\n";
print "nSuccNonZeroReal/nNonZeroReal \t= $nSuccNonZeroRea1/$nNonZeroReal \t=
\$accNonZeroReal\n";
print "nSuccNonZeroEff/nNonZeroEff \t= $nSuccNonZeroEff/$nNonZeroEff \t=
\$accNonZeroEff\n".
}
print "\ncoverage:\n";
print "nNonZero/nTotal = $nNonZero/$nTotal = \$cvgNonZero\n";
print "nNonNull/nTotal = $nNonNull/$nTotal = \$cvgNonNu11\n";
print "nCand/nTotal = $nCand/$nTotal = \$cvgCand\n";
print RES "nSuccTotal/nTotal \t= $nSuccTotal/$nTotal \t= \$accTotal\n";
print RES "nSuccCand/nCand \t= $nSuccCand/$nCand \t= \$accCand\n";
print RES "nSuccNonNull/nNonNull \t= $nSuccNonNull/$nNonNull \t= \$accNonNull\n";

```
```

print RES "nSuccNonZero/nNonZero \t= $nSuccNonZero/$nNonZero \t= \$accNonZero\n";
print RES "nSuccZero/nZero \t= $nSuccZero/$nZero \t= \$accZero\n";
print RES "nSuccNull/nNull \t= $nSuccNull/$nNull \t= \$accNull\n";
print RES "nSuccCandNull/nCandNull = $nSuccCandNull/$nCandNull \t= \$accCandNu1l\n";
if( \$optsHash{'e'} ){
print RES "\n";
print RES "nSuccNonZeroReal/nNonZeroReal \t= $nSuccNonZeroReal/$nNonZeroReal \t=
\$accNonZeroReal\n";
print RES "nSuccNonZeroEff/nNonZeroEff \t= $nSuccNonZeroEff/$nNonZeroEff \t=
\$accNonZeroEff\n";
}
print RES "\ncoverage:\n";
print RES "nNonZero/nTotal = $nNonZero/$nTotal = \$cvgNonZero\n";
print RES "nNonNull/nTotal = $nNonNull/$nTotal = \$cvgNonNul1\n";
print ReS "nCand/nTotal = $nCand/$nTotal = $cvgCand\n";
print ACC "$accTotal\n";
close(RES);
close(ACC);
\#\#\#\#\#\#\#\#\#\#\#\#\#\#end main section\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#\#
\#\#subroutines****************************************************************
\#stemMatch( stem, word) means is stem the stem of word
\#do simple plurals and ing (for eg "playing")
sub stemMatch{
my (\$stem, $word) = @_;
    return 1 if($stem eq \$word );
\#else attempt to pull off plural and then compare again
if( $word =~ /^(.+)s$/ ) {
return 1 if( \$stem eq \$l );
}
if( $word =~ /^(.+)es$/ ){
return 1 if( \$stem eq \$1 );
}
if( $word =~ /^(.+)ies$/ ){
return 1 if( \$stem eq (\$1 . "Y") );
}
if( $word =~ /^(.+)ing$/ ) {
return 1 if( \$stem eq \$1 );
}
return 0;
}
sub zeroDefaultScore{
return \&defaultScore("z");
}
sub nullDefaultScore{
return \&defaultScore("n");
}
sub defaultScore{
my $inputStr = shift;
    if($defScoreArg ne n"){
if($defScoreArg eq "def0"){
        return '-';
    }
    if($defScoreArg eq "def1"){
return '+';
}
\#def2 let fall through
if($defScoreArg eq "def3"){
        if($inputStr eq "z"){
if( \$zflip eq '+' ){
\$zflip = '-';
return \$zf1ip;
}
else{
\$zflip = '+';
return \$zflip;
}

```
```

                } # else n
                    if( $nflip eq '+' )(
                    $nflip = '-';
                    return $nflip;
                }
                else{
                    $nflip = '+';
                    return $nflip;
                }
            }
    }
    if( $butDefaultFlip && $butFlag ){ #flip prev score
        if( $prevScore eq '+' )(return '-';}
        else{return '+';}
    }
    else{
        return $prevScore;
    }
    }
sub isCandidateString{ \#applies to both sentences and single words
my \$s = shift;
my $ans = "n;
    if( ($optsHash{'j'} \&\& (\$s =~ /\:J/)) )(
\$ans = $ans . "J";
    }
    if( ($optsHash{'n'} \&\& (\$s =~ ハ:N/)) ){
\$ans = $ans . "N";
    }
    if(($OptsHash{'v'} \&\& (\$s =~ 八:V/)) ) (
\$ans = $ans . "V";
    }
    if( ($optsHash{'r'} \&\& (\$s =~ ハ:R/)) )(
\$ans = \$ans . "R";
}
return \$ans;
}
sub isNegWord{
my \$s = shift;
\$s =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists($negHash{\$w}) ){
return 1;
}
else{
return 0;
}
}
sub isButWord{
my \$s = shift;
\$s =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists($butHash{\$w}) ) {
return 1;
}
else{
return 0;
}
}
sub isDeflagWord{
my \$s = shift;
\$s =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists($deflagHash(\$w}) ){
return 1;
}
else{
return 0;

```
```

    }
    }
sub isContrastWord{
my \$s = shift;
\$s =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists($contrastHash{\$w}) ) {
return 1;
}
else{
return 0;
}
}
sub isModalWord{
my \$s = shift;
\$S =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists($modalHash{\$w}) ) {
return 1;
}
else{
return 0;
}
}
sub isPresupWord{
my \$s = shift;
\$s =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists($presupHash{\$w}) ) {
return 1;
}
else{
return 0;
}
}
sub isExcessWord{
my \$s = shift;
\$s =~ /^([^\:]+)\:/; \#grab word
my \$w = $1;
    if( exists ($excessHash{\$w}) ) {
return 1;
}
else{
return 0;
}
}
\#\#\#END
===
\#!/home/bin/perl
\#123456789 123456789 123456789 123456789 123456789 123456789 123456789 123456789
\#\#\#filename: run-5-score-corp.pl
\#\#\#author: Adam Longton, longton@cs.ubc.ca

# this script runs 5-score-corp.pl across multiple corpora and summarizes the results.

# if WordNet expansion was used, switch on the commented out lines tagged with "\# use

this for WordNet version"
use strict;
use warnings;
my \$usage = "usage: perl \$0 opts scoredLexiconFile\n"; \#this is for non wN, use the
scored lex file directly.
\#my \$usage = "usage: perl \$0 opts pathofScoredCandsFilesFromPh3\n"; \# use this for
WordNet version
die $usage if ($\#ARGV < 0);
my $opts = "";
if($\#ARGV > 0){
\$opts = shift;
}
my \$lex = shift;

```
```


# these paths to be setup for a particular experimental environment

my \$bline = "/.autofs/homes/ubccshome/l/longton/adam/proj";
my $scripts = "$bline/exp/scripts";
my $here = "$bline/exp"; \#experiment root
my @corps = ("apex","canon","creative","nikon","nokia");
my \$line; my \$accavg = 0; my \$accadd = 0; my \$n = 0;
my \$succSum = 0; my \$totalSum = 0; my $fracAvg = 0;
open(ACCS,">accs$opts.txt") or die "error: couldn't open accs\$opts.txt\n";
print ACCS "command: perl \$0 \$opts \$lex\n"; \# track how the accs output file was
generated
my \$corp;
foreach \$corp (@corps){
`perl $scripts/5-score-corp.pl $opts $lex $here/corp/1out-$corp.txt > tmp5out- $corp$opts.txt`;
\#'perl \$scripts/5-score-corp.pl \$opts $lex/scoredAdjs-$corp.txt \$here/corp/1out-
$corp.txt > 5out-$corp\$opts.txt`; # use this for WordNet version     open(ACC,"acc.txt") or die "error: couldn't open acc.txt for $corp\n";     $line = <ACC>;     close(ACC);     chomp $line;     $accadd += $line;     $n++;     open(RES,"results-$corp$opts.txt") or die "error: couldn't open results- $corp$opts.txt\n";     $line = <RES>;     $line = <RES>; #second line has the acc     close(RES);     chomp $line;     print ACCS "$corp: $line\n";         `rm acc.txt`;
\# add up the nSuccs and the n's to get a weighted fractional avg acc too:
if(\$line =~ m| (\d+)/(\d+)|){
\$succSum += \$1;
\$totalSum += $2;
    }
}
if($n>0){\$accavg = $accadd/$n;}
if($totalSum>0) {$fracAvg = sprintf("%.7f", $succSum/$totalSum);} \#jy08
print ACCS "Avg acc: \$accavg, weighted: $succSum/$totalSum = \$fracAvg\n";

# also run on concatenation of the subcorpora for comparison

\$corp = "hu";
`perl $scripts/5-score-corp.p1 $opts $lex $here/corp/lout-$corp.txt > tmp5out- $corp$opts.txt`;
\#`perl $scripts/5-score-corp.pl $opts $lex/scoredAdjs-$corp.txt $here/corp/lout-$corp.txt > 5out-$corp$opts.txt`; \# use this for WordNet version
open(RES,"results-$corp$opts.txt") or die "error: couldn't open results-$corp$opts.txt\n";
\$line = <RES>;
\$line = <RES>;
close(RES):
chomp $line;
print ACCS "$corp: \$line\n";
`rm acc.txt`;
close(ACCS);
`rm tmp5*`; \#cleanup all the big scored sentence files
\#\#END
===
===

```
```

