SIMULTANEOUS GENERATION OF DOMAIN-SPECIFIC LEXICONS FOR MULTIPLE SEMANTIC CATEGORIES

by

Michael Wayne Thelen

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing

The University of Utah

December 2001

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Michael Wayne Thelen

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Ellen Riloff

Cynthia Thompson

William Thompson

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Michael Wayne Thelen in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Ellen Riloff
Chair, Supervisory Committee

Approved for the Major Department

Tom Henderson
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

Semantic lexicons are useful for many tasks in natural language processing. Until recently, semantic lexicons were built by hand, which required many hours of effort. Techniques have been developed to automate this process, but precision of automatically-generated semantic lexicons has generally been low. I have developed an algorithm, BASILISK, that exploits the tendency of words to belong to a single semantic category within a limited domain, as well as the tendency of extraction patterns to extract semantically similar words. BASILISK uses two original techniques to improve performance: application of statistics gathered over a large body of extraction patterns, and generation of semantic lexicons for multiple categories simultaneously. BASILISK has been used to generate semantic lexicons for six categories using the MUC-4 terrorism corpus, achieving substantially better precision than achieved by previous techniques.

To Kristen

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Why Do We Need Semantic Lexicons?

Suppose you are the head of a major corporation that wants to keep on top of late-breaking news. If there have been any corporate mergers or acquisitions in the past 24 hours, you would like to know about them right away. Or perhaps you are working for a government anti-terrorism department, and you need immediate information about all perpetrators of terrorist activity in the past five years. Or maybe you are developing an automatic question-answering system and would like to be able to have your system answer questions like, "Who was the mayor of Billings, Montana in 1965?"

What do all these tasks have in common? They can all benefit from natural language processing. The first task might be handled by a program that crawls the World Wide Web, picking up and summarizing articles relevant to corporate activity. The second task might be addressed by a program that reads through newswire articles about terrorism, extracting the names of perpetrators of terrorist acts. For the third task, the problem could be solved by mapping the question into a query for a database of historical information.

As well as profiting from natural language techniques, these applications may also benefit from semantic information. Semantic information deals with meanings of words. It is often useful to know semantic information about the entity or action represented by a word, in addition to the word's syntactic role in a sentence. For example, in the question above, "Who was the mayor of Billings, Montana in 1965?" the question-answering program will need to know that "the mayor" is a phrase referring to a human entity, "Billings" refers to a city, "Montana" is the name of a state, and "1965" is an expression of time. Semantic information can be represented in many ways, but one of the simplest and most straightforward representations is a *semantic lexicon*.

There are different kinds of semantic lexicons; however, for our purposes, the phrase *semantic lexicon* will refer to a list of words, each with a single associated semantic category. For example, a semantic lexicon might specify the word "mayor" as belonging to the semantic class *human*. A semantic lexicon may be used in conjunction with a semantic hierarchy, so that information can be gained not only from the direct semantic designation of each word, but also from the hierarchy both above and below that specific designation. In the present example, one might construct a hierarchy in which the semantic categories *city* and *state* are both subordinate to the semantic category *location*, in which case both "Billings" and "Montana" could be recognized as belonging to the class *location* as well.

These brief examples are only a few of the tasks for which semantic information can be useful. There are many other applications for which semantic information has been observed to improve performance. For example, semantic information has been shown to help information extraction systems, such as PALKA [11], CRYSTAL [25], and Riloff & Schmelzenbach's IE system [20]. Semantics are also used in anaphora resolution systems, such as RESOLVE [15], MLR [2], Kameyama's system [10], and Ge, Hale, & Charniak's system [8]. For the recent TREC Question Answering task [28], several of the top-scoring systems used heavy semantic typing of questions and answers. These systems include InfoXtract [26], LASSO [17], and AT&T's system [24]. Reading comprehension systems have also been shown to benefit from semantic information, as evidenced by Deep Read [9], the systems of Charniak *et al.* [4], and QUARC [22].

QUARC was a system I designed for the task of reading comprehension, and it became the impetus for this research. In the reading comprehension task, a system reads simple texts and then answers questions about the texts to test its understanding. QUARC used hand-coded rules to select the best answer for each question. I developed these rules through extensive examination of the training data, and many of the rules used semantic information from *human*, *location*, and *time* semantic categories. QUARC was able to achieve a score of 40% on a particular set of reading comprehension tests, which was better than previous results. By isolating rules that looked for semantic information, I was able to determine that the *who*, *when*, and *where* questions all benefitted greatly from semantic knowledge. With more and better semantic knowledge, reading comprehension systems such as QUARC may be able to improve well beyond their limited capabilities of today.

1.2 The Problem with Previous Approaches

1.2.1 Broad-coverage Lexicons

Since semantic lexicons are so useful in a variety of tasks, one might expect that someone somewhere has undertaken a project to create the ultimate comprehensive semantic lexicon, which would obviate the need for semantic lexicon generation techniques. In fact, there have been several large projects undertaken, such as WordNet [16] and CYC [12]. Both of these resources are useful for the researcher who requires semantic information.

Broad-coverage resources such as these contain thousands of words with part-of-speech and semantic information, but they are often insufficient to cover the idiomatic vocabulary that is particular to specific domains [23]. For example, in the MUC-4 terrorism corpus, the word "department" frequently refers to a geographic designation. However, in general English usage, the word "department" is rarely used to designate a geographic location. If a word can have multiple parts-of-speech or word senses, the sheer amount of information in a broad-coverage resource can actually increase the difficulty of disambiguating its meaning. On the other hand, domain-specific lexicons have the advantage of highly constraining the natural ambiguity of language. For example, a lexicon specific to the terrorism domain may safely omit most of the meanings of the word "shot" that are irrelevant to terrorism, such as the throw of a basketball, a drink of hard liquor, or an injection with a needle. In the terrorism domain, "shot" almost always refers to the shooting of ammunition from a weapon.

1.2.2 Manually-generated Lexicons

Automatic generation of semantic lexicons is also a worthwhile pursuit in terms of saving time and human effort. Until very recently, semantic lexicons were painstakingly compiled by hand. Depending on the task for which a lexicon is intended, building it manually can take anywhere from a few hours to a few months. In addition, even the most meticulously-constructed semantic lexicon is prone to errors of omission. Language is so diverse that there will always be valid words and phrases that the lexicon creator simply did not know or forgot to include. For example, "M-60" is a type of machine gun mentioned fairly often in terrorism newswire articles, and would be a good word for a *weapon* semantic lexicon. However, it is likely that someone compiling such a lexicon by hand would forget or be unfamiliar with this weapon.

For these reasons, it is important to develop methods for building domain-specific semantic lexicons automatically. It is also important to develop techniques for making

these semantic lexicons as accurate as possible, to minimize the amount of time that a human will have to spend reviewing the automatically-generated lexicons for use in real-world applications. The research presented in this thesis addresses these problems.

1.3 A New Approach

The main result of my research is an algorithm that generates high-quality, domain-specific semantic lexicons automatically. My algorithm, BASILISK (Bootstrapping Approach to SemantIc Lexicon Induction using Semantic Knowledge), takes advantage of two main observations. The first observation is that extraction patterns[1] tend to extract semantically similar words, and that statistics over a variety of extraction patterns can provide reliable evidence that a word belongs to a semantic category. The second observation is that words tend to be used in a single word sense within a limited domain. This observation can be exploited by generating semantic lexicons for multiple semantic categories simultaneously, using evidence from all categories to determine the proper categorization for any particular word.

1.3.1 Collective Evidence Over a Body of Extraction Patterns

Since extraction patterns tend to extract semantically related words, collective evidence gathered from a large number of extraction patterns may reliably identify words that belong to a semantic category.

The observation that extraction patterns tend to extract semantically similar words was originally utilized by Riloff & Jones in their meta bootstrapping algorithm [19]. Figure 1.1 shows a real example of an extraction pattern from the terrorism domain. One can see that the pattern "was killed in <X>" extracts "Peru", "Colombia", "eastern El Salvador" and other *location* phrases. The heart of the meta bootstrapping algorithm is an inner level of bootstrapping, *mutual bootstrapping*, that exploits this tendency at each iteration by finding the best extraction pattern for a semantic category and adding all of its extractions to the lexicon for that category. An outer layer of bootstrapping then attempts to choose the best words from the ones that were added during the mutual

[1] An extraction pattern is a syntactic pattern that usually consists of a *trigger phrase* and a syntactic *slot* that is filled when the trigger phrase is found in a text. For example, in the extraction pattern "<X> killed", the trigger phrase is "killed" in the active voice, and the slot to be filled is the syntactic subject of the verb. The phrase that fills the slot is called the *extraction*.

bootstrapping iterations. Meta bootstrapping uses this multi-level approach to generate semantic lexicons and lists of relevant extraction patterns.

"was killed in <X>" extracts:

- "Peru" (*location*)
- "this action" (*event*)
- "clashes" (*event*)
- "Urrao" (*location*)
- "this sector" (*location*)
- "a shootout" (*event*)
- "San Francisco" (*location*)
- "eastern El Salvador" (*location*)
- "Colombia" (*location*)
- "San Salvador" (*location*)

Figure 1.1. Extraction Pattern Example

Unfortunately, mutual bootstrapping can introduce many incorrect lexicon entries as well as legitimate ones, because even extraction patterns that are highly correlated with a semantic category may extract non-category phrases as well. For example, the extraction pattern "was killed in <X>" also extracts non-*location* phrases such as "a shootout", "clashes", and "this action". Infecting the lexicon with non-*location* words such as these can have a drastically negative effect on the bootstrapping process. For this reason, BASILISK is wary of trusting any individual extraction pattern too much when selecting words for the lexicon.

While meta bootstrapping is pattern-centric in that it relies heavily on finding the single best extraction pattern for a semantic category, BASILISK is word-centric in that it seeks to choose the best words for a semantic category by using statistics over all extraction patterns. Using information from many extraction patterns increases precision

of the semantic lexicon by requiring credible evidence from a variety of sources before a candidate word is added to the lexicon.

To continue the example above, by accounting for extraction patterns in addition to "was killed in <X>", BASILISK may see that the words "shootout", "clashes", and "action" do not tend to be extracted by any other *location*-related extraction patterns. On the other hand, true *location* words such as "Peru" and "Colombia" will be extracted repeatedly by other *location*-related patterns. In this way, BASILISK can determine that words consistently extracted by patterns associated with a category have a high likelihood of belonging to that category.

1.3.2 Bootstrapping Multiple Categories Simultaneously

In addition to using collective evidence over a large body of extraction patterns, bootstrapping multiple categories simultaneously can also improve the precision of the semantic lexicon for all categories.

It is possible to benefit from generating multiple categories simultaneously because of a "one sense per domain" assumption, which says that a word tends to be used with a single word sense within a limited domain. This assumption is based on the "one sense per discourse" observation made by Gale, Church, & Yarowsky [7]. They noticed that within a single text, any given word has a strong tendency to be used only in a single sense. For example, if a text contains the word "plant" in reference to a living organism, it is highly improbable that the word "plant" will be used in the same text to refer to a manufacturing facility. Since texts within a limited domain comprise a loose discourse, this theory can be broadened into the "one sense per domain" assumption. In a practical sense, this assumption allows BASILISK to assert that within a limited domain, each word may only belong to a single semantic category. This assertion is not always strictly true, but it is true of most words most of the time.

Previous semantic lexicon generation algorithms have focused on generating lexicons for one semantic category at a time. BASILISK is able to exploit the "one sense per domain" assumption by generating semantic lexicons for multiple categories simultaneously. For example, an algorithm that only considers a single semantic category may incorrectly hypothesize "shootout" as a *location* word because it occurs in the *location*-related extraction pattern "was killed in <X>". In contrast, BASILISK may be able to use information from the *event* category to determine that it should be a *event* word instead. This may happen if the word "shootout" is extracted by several *event*-related patterns.

If "shootout" is added to the lexicon for the *event* category, then BASILISK will never consider it as a candidate for the *location* category again. Even if "shootout" is not added to the *event* lexicon, the fact that it occurs in several *event*-related patterns will discourage BASILISK from classifying it as a *location*.

BASILISK has been used to generate semantic lexicons for six categories in the MUC-4 terrorism corpus: *building*, *event*, *human*, *location*, *time*, and *weapon*. Using collective evidence over many extraction patterns improved accuracy for each semantic category, in most cases substantially. In addition, generating semantic lexicons for multiple categories simultaneously improves the accuracy of the BASILISK algorithm and of meta bootstrapping for some categories. Even greater improvements are achieved by taking advantage of both techniques at the same time.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 describes related work in automatic semantic lexicon generation and unsupervised bootstrapping, which is the general approach used by BASILISK. Chapter 3 describes the BASILISK algorithm in detail and presents results showing its improvement over meta bootstrapping. Chapter 4 describes the multiple-category versions of meta bootstrapping and BASILISK, and presents results showing improvement over their single-category versions. Chapter 5 presents an analysis of BASILISK's performance compared to previous techniques, and summarizes the conclusions that can be drawn from this research.

CHAPTER 2

RELATED WORK

My research focuses on the simultaneous generation of semantic lexicons using unsupervised bootstrapping techniques. There has been much work done previously, both in the areas of semantic lexicon generation and unsupervised bootstrapping. In this chapter, I will summarize some of the previous work that is best-known and most closely related to my research.

2.1 Semantic Lexicon Generation

2.1.1 Riloff & Shepherd, 1997

One of the first algorithms for automated acquisition of semantic lexicons from unannotated text was proposed by Riloff & Shepherd [21]. The algorithm is based on the observation that words in a semantic category often co-occur with each other within certain syntactic constructs. The examples cited are conjunctions (*lions and tigers and bears*), lists (*lions, tigers, bears, ...*), appositives (*the stallion, a white Arabian*), and nominal compounds (*Arabian stallion; tuna fish*). One can take advantage of this observation to create a semantic lexicon by a bootstrapping process.

The process begins with a few seed words for the semantic category. The seed words are chosen by a human. All instances of these seed words as head nouns[1] in the corpus are found, and co-occurrences with these words are considered within a context window. The window includes one word to the left and right of a seed word. Non-seed words within these context windows are given a score based on the ratio of each word's occurrence within context windows to that word's occurrence throughout the corpus. Next, stopwords and numbers are removed, and any word that occurred fewer than six times in the corpus is removed from consideration. All non-seed words are ranked based upon their score, and the five top-ranked nouns are added to the seed word list. Beginning

[1] A head noun is the word that occurs as the "head" of a noun phrase (NP). It is usually the rightmost word in the NP. For example, the head noun in the NP "the white rabbit" is the word "rabbit".

with this new, larger seed word list, the bootstrapping loop iterates again. At the end of a certain number of iterations, the algorithm outputs is a ranked list of words that are hypothesized to belong to a certain semantic category. To determine which of these words are valid semantic category members and which are not, this list must be filtered by a human.

Rankings of the top 200 words for each category yielded an average of about 35 true category members and about 25 subparts of category members for *vehicle* and *weapon* categories. This constitutes a limited degree of success. This algorithm was among the first to attempt automatic semantic lexicon generation, but the precision (about 17%) of the final semantic lexicons was fairly low.

2.1.2 Roark & Charniak, 1998

Another algorithm for acquiring semantic lexicons from unannotated text is given by Roark & Charniak [23]. This algorithm was inspired by the Riloff & Shepherd approach. It shares the basic observation that members of a semantic category tend to occur together within certain syntactic constructs. Their approach focuses exclusively on these syntactic constructs, using complete parse trees of each sentence in the corpus to identify lists, appositives, conjunctions, and compound nouns.

As in the Riloff & Shepherd approach, the process begins with a small list of seed words. These words are chosen from among the most frequently-occurring head nouns in the corpus to ensure broad coverage. The algorithm proceeds by looking within the syntactic constructs throughout the corpus to find non-seed words that co-occur with seed words. All nouns with the highest score are added to the seed word list and the process is repeated. After a number of iterations (the authors used 50), all remaining non-seed words are discarded from consideration. The seed word list is then returned to its original members, and a final ranking stage begins, using a log-likelihood metric for ranking. This ranking loop iterates for the same number of times that the first bootstrapping loop was run, adding words to the seed word list, and finally the algorithm outputs two lists. The first is a list of head nouns, ranked by when they were added to the seed word list in the ranking loop. The second is a list of compound nouns from the corpus, ordered by when their head nouns were added to the seed word list.

This algorithm was run for various semantic categories over the MUC-4 terrorism corpus [1] and the Penn Treebank Wall Street Journal corpus [13]. This new approach yielded, on average, 2–3 times the number of valid category words found by the Riloff

& Shepherd approach, with approximately twice the level of precision. This algorithm performed with 33% precision on the *vehicle* category and 36% precision on the *weapon* category.

2.1.3 Multi-Level Bootstrapping: Riloff & Jones, 1999

Multi-level bootstrapping [19] seeks to automate the acquisition of semantic lexicons and extraction patterns, given only a large corpus of unannotated texts relevant to a specific domain.

The Riloff & Jones algorithm consists of two levels of bootstrapping. The inner level is called *mutual bootstrapping* because the semantic lexicon and extraction patterns are able to bootstrap off each other; The outer level is called *meta bootstrapping* because it is a bootstrapping process upon the mutual bootstrapping process.

Empirical results from this process indicate that it can produce more accurate semantic lexicons than previous lexicon generation techniques. My research is similar to this work. In fact, part of my research entails enhancing the meta bootstrapping algorithm to improve its performance. For this reason, meta bootstrapping is explained in greater detail in Section 3.1.

2.1.4 Other Approaches

In addition to generating lists of words that belong to semantic categories, there are other approaches to acquiring semantic information automatically.

One method for generating semantic lexicons automatically is presented by Thompson & Mooney [27]. In this context, a "semantic lexicon" refers to phrases paired with formal meaning representations in Prolog. Their system, WOLFIE, generates a lexicon of meaning representations given a corpus of sentences paired with semantic representations. It begins by assigning general candidate meanings for each phrase, iteratively constraining phrase meanings until the learned semantic lexicon covers the entire corpus or until phrase meanings can be constrained no further. WOLFIE's effectiveness was measured by using its learned lexicons in a semantic parser, CHILL [31], and measuring CHILL's performance as an interface to a database of geography facts. Hand-crafted lexicons and other automatically generated lexicons were also used for comparison. Overall, WOLFIE performed comparably to hand-built lexicons for this task.

Another algorithm for obtaining semantic information is given by Ge, Hale, & Charniak [8], who present a probabilistic model for anaphora resolution. Their anaphora

resolution algorithm incorporates four main factors, one of which is semantic information about the gender and animaticity of certain words. The authors devised a method for automatically acquiring this information using the simple co-occurrence of nouns and pronouns in a text corpus. They employ the log-likelihood ratio [6] using the raw frequencies of each pronoun class (masculine, feminine, neuter) in the corpus, as well as the co-occurrence statistics of nouns that are likely referents to those pronouns. This algorithm is able to learn gender information with approximately 70-90% precision.

Caraballo presents an algorithm for automatically constructing a semantic hierarchy. Her algorithm takes advantage of the observation that semantically related words often occur within conjunctions and appositives, which was utilized by Riloff & Shepherd [21] and Roark & Charniak [23]. Caraballo's method uses bottom-up clustering techniques in order to build a binary hierarchy of hypernym relationships. Following the example of WordNet [16], word A is said to be a hypernym of word B if native English speakers would accept the sentence, "B is a (kind of) A." A portion of the final hierarchy was evaluated as achieving precision of 30-60% for the hypernym relationships.

2.2 Unsupervised Bootstrapping

2.2.1 Word Sense Disambiguation: Yarowsky, 1995

Many words have different meanings in different contexts. Meanings that are sufficiently distinct are called *word senses*. For example, the word "plant" may be used generally to refer to a living organism or a factory, among other possible uses. Unfortunately, the disambiguation of such polysemous words[2] is not always straightforward. David Yarowsky has developed an unsupervised bootstrapping algorithm that performs word sense disambiguation with high accuracy [30].

There are two main hypotheses behind the bootstrapping algorithm. The first is a "one sense per collocation" hypothesis. This states that words have a strong tendency to exhibit only one sense in a given collocation. For example, the word "plant" may have two major word senses, but when found in collocation with the word "life", it almost always takes on the "living organism" sense. This phenomenon was observed and quantified by Yarowsky [29]. The second hypothesis is a "one sense per discourse" observation.

[2] A word is called "polysemous" if it has more than one word sense. Polysemous words are more common in natural language than one might expect, which is why word sense disambiguation is an important problem.

Gale, Church, and Yarowsky observed that words strongly tend to take on a single word sense within a given discourse or document [7]. Yarowsky's algorithm uses these two hypotheses throughout an iterative bootstrapping procedure that trains classifiers for binary-ambiguous words (i.e. words with two major senses).

The algorithm begins with a small list of seed collocations provided by a human. For example, a human might give the "plant" classifier the word "life" as a collocation for word sense A, and "manufacturing" as a collocation for word sense B. The algorithm uses a log-likelihood ratio to label instances of the word "plant" about which it is most certain. Based on these newly-classified words, the algorithm discovers new collocations that are indicative of each word sense. The best collocations are added to the seed collocation list, and the loop repeats. If evidence is strong, the one-sense-per-discourse constraint may be invoked to classify unresolved words, or to correct mistakes of previous iterations. For example, if 5 instances of "plant" within a document exhibit word sense A, and the sixth is unknown or labeled with word sense B, then the "one sense per discourse" heuristic may change the classification of the sixth instance to word sense A.

Beginning with only these two heuristics and a very small seed collocation list for each ambiguous word, Yarowsky's algorithm is able to achieve a very high success rate, averaging over 96% for 12 binary-ambiguous words.

2.2.2 Co-training: Blum & Mitchell, 1998

For many machine learning tasks, there is a very large amount of unlabeled data available, but labeled data is more scarce. Co-training [3] seeks to take advantage of this situation by building classifiers from a small amount of labeled data and a large amount of unlabeled data.

For a binary classification task, co-training has been theoretically proven to improve the performance of individual classifiers, given certain assumptions. One must be able to describe each training instance in terms of two independent kinds of information. The authors give the example of a web page, which can be described in terms of its actual content or the text of the hyperlinks that point to it. It must also be assumed that each kind of information is sufficient to classify an instance. For example, given either the content of a web page or the text of hyperlinks pointing to it, one must be able to make a classification about that web page (e.g., whether the web page is the homepage of a university professor). Co-training operates within this framework of two "views" of training instances, each view being sufficient for classification. Incidentally, Blum &

Mitchell also suggest that Yarowsky's word sense disambiguation work can be seen as a co-training framework, with one view based on the "one sense per collocation" hypothesis and the other view based on the "one sense per discourse" hypothesis.

Co-training takes as input a small amount of labeled training examples and a large amount of unlabeled training examples. One classifier is trained for each view of the data. Each of these classifiers is allowed to classify p positive examples and n negative examples from the unlabeled data. These newly-labeled examples are added to the labeled data, and the classifiers are re-trained. In this way, the most confidently-labeled examples of one classifier can be used as training data for the other, and both classifiers grow more accurate as a result.

The authors show that given this binary classification task and the fulfilled prerequisites described above, co-training will learn to classify new instances with accuracy. This conclusion is affirmed by empirical results.

2.2.3 EM and Active Learning: McCallum & Nigam, 1998

McCallum & Nigam [14] use the Expectation-Maximization (EM) algorithm [5] to enhance active learning for text classification. EM is an unsupervised iterative bootstrapping algorithm for predicting missing data values and estimating a generative model for certain problems. Given a model and data with missing values, EM estimates the missing values based on the model, then reestimates the model based on the hypothesized values.

In this context, active learning is used for classifying texts by choosing unlabeled texts to be presented to a user for the correct classification. Since such requests are considered to be expensive, the algorithm tries to choose texts that would yield the most information if their classification were known. McCallum & Nigam find such texts by using a Query By Committee (QBC) algorithm, which creates various classifiers over the training data and then records their classifications of unlabeled texts. Texts on which the classifiers disagree strongly are displayed to the user for classification.

McCallum & Nigam enhance the active learning algorithm by using EM to train the QBC classifiers before their classifications are used to calculate disagreement. They also enhance the QBC method by using "distance-weighted pool-based sampling" to choose unlabeled texts with high disagreement. This sampling allows the algorithm to choose from all unlabeled texts, while encouraging the algorithm to choose from a densely populated area of the search space for maximum relevance.

In experimental results, QBC with pool-based sampling reduces the need for labeled training documents by 42% over previous QBC methods. In addition, combining pool-based sampling with EM allows the algorithm to learn with only 58% as many labeled texts as EM alone, and only 26% as many as QBC alone. These results show that employing EM and active learning can greatly reduce the need for labeled training data in text classification tasks.

CHAPTER 3

DESCRIPTION OF ALGORITHMS

One of my main claims is that using statistical information over many extraction patterns improves semantic lexicon generation. This idea was originally conceived to overcome a problem with meta bootstrapping: that it relies heavily on individual extraction patterns that are not completely reliable. I developed a new algorithm, BASILISK, to correct this weakness by utilizing collective evidence over many extraction patterns. This chapter describes the meta bootstrapping and BASILISK algorithms in detail. The other main claim of this thesis, that generating lexicons for multiple semantic categories simultaneously can improve precision, will be addressed in Chapter 4.

3.1 Meta Bootstrapping

Meta bootstrapping is an algorithm developed by Riloff & Jones for simultaneously generating lists of extraction patterns and noun phrases associated with a semantic category. As explained in Section 2.1.3, meta bootstrapping consists of two levels of bootstrapping. The inner level is called *mutual bootstrapping* because the semantic lexicon and extraction patterns bootstrap off each other, and the outer level is called *meta bootstrapping* because it is built upon the inner bootstrapping layer.

Learning begins with a small set of human-chosen seed words for the semantic lexicon. All extraction patterns throughout the corpus are found. These patterns are ranked according to the *RlogF* metric used by AutoSlog-TS [18]. Using this function, the score for each extraction pattern is

$$RlogF(pattern_i) = \frac{F_i}{N_i} * \log_2(F_i) \tag{3.1}$$

where F_i is the number of unique lexicon entries extracted by $pattern_i$ and N_i is the total number of unique phrases extracted by $pattern_i$. All the noun phrases extracted by the highest-ranked extraction pattern are added to the seed word list, the extraction pattern is added to a "best patterns" list, and the mutual bootstrapping process repeats.

While this inner bootstrapping layer works well on its own, it is also subject to noise. If an extraction pattern tends to extract good words for the semantic lexicon but also extracts bad words, then the semantic lexicon will become polluted and performance will degrade quickly. To slow the degradation of performance, the meta bootstrapping layer was added.

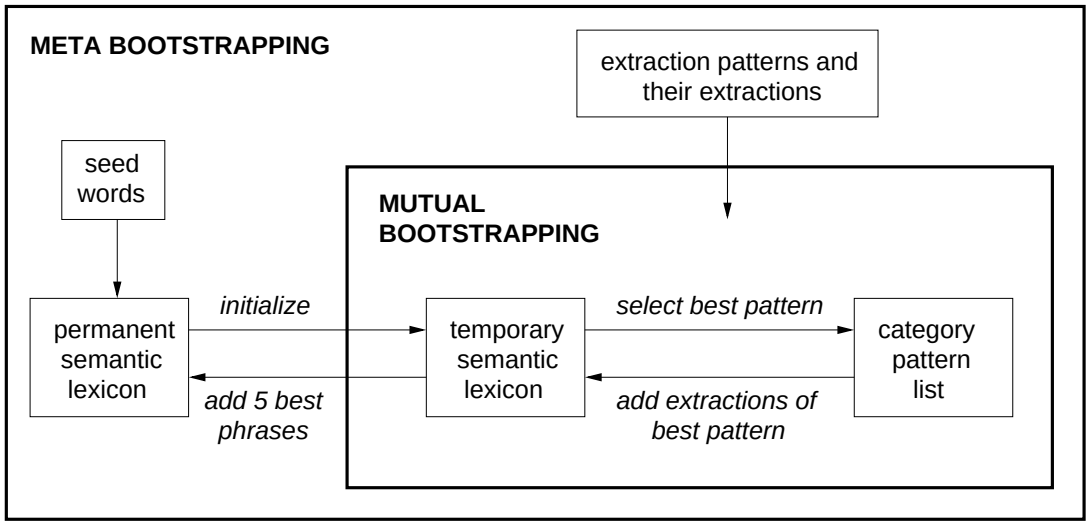

Figure 3.1. Meta Bootstrapping Algorithm

With the meta bootstrapping layer added, mutual bootstrapping proceeds normally except that it adds candidate phrases to a temporary lexicon. After a number of iterations of mutual bootstrapping, the five most reliable lexicon entries in the temporary lexicon are added to a permanent semantic lexicon, and all the others are thrown away. These reliable lexicon entries are chosen by ranking all candidate phrases according to how many extraction patterns in the "category pattern" list extracted each one. The scoring function also includes a tie-breaking factor that accounts for the individual strength of each pattern. The function used is

$$score(word_i) = \sum_{k=1}^{N_i} 1 + (0.01 * RlogF(pattern_k)) \tag{3.2}$$

where N_i is the number of extraction patterns in the "category pattern" list that extracted $word_i$, and $RlogF(pattern_k)$ is the $RlogF$ scoring function described by Equation 3.1.

After adding the five most reliable candidate phrases to the permanent lexicon, the temporary lexicon and category pattern list are cleared, and the mutual bootstrapping

- Generate all extraction patterns in the corpus and save the patterns with their extractions
- *perm_lexicon* = {seed words}
- META BOOTSTRAPPING
  1. *temp_lexicon* = {}
  2. *category_patt_list* = {}
  3. MUTUAL BOOTSTRAPPING (10 iterations)
     (a) Score all extraction patterns according to *RlogF*
     (b) *best_patt* = top ranked pattern not already in *category_patt_list*
     (c) Add *best_patt* to *category_patt_list*
     (d) Add extractions of *best_patt* to *temp_lexicon*
     (e) Go to Step 3a
  4. Score candidate words in *temp_lexicon*
  5. Add top 5 candidate words to *perm_lexicon*
  6. Go to Step 1

Figure 3.2. Meta Bootstrapping Pseudo-Code

process begins anew. Meta bootstrapping may be run for as many iterations as the user likes, adding five lexicon entries at each iteration. When processing stops, the multi-level bootstrapping will have produced a semantic lexicon and a list of extraction patterns relevant to the semantic category.

Although meta bootstrapping generates semantic lexicons with greater precision than previous automatic semantic lexicon generation techniques, it does have one major shortcoming. Meta bootstrapping is built upon the mutual bootstrapping algorithm, which trusts the best individual category patterns to extract only semantically valid category words. This assumption is often false, which can cause non-category words to infiltrate the lexicon. For example, the "was killed in <X>" extraction pattern presented in Section 1.3 extracts several valid *location* words, but it also extracts other words such as "shootout" and "clashes". If "was killed in <X>" were the best *location* pattern, these spurious words would be added to the lexicon.

The meta bootstrapping layer alleviates this problem somewhat; however, non-category

words can still be added to the permanent lexicon if they are extracted by a few category patterns, or even merely by the single best category pattern. For example, after a certain number of meta bootstrapping iterations, it is often the case that every candidate word is only extracted by one "category pattern". When this happens, the algorithm has no choice but to choose words for the lexicon based on their extraction by a single pattern. In this case, the tie-breaking factor in Equation 3.2 causes the extractions of the single best pattern to be added to the permanent lexicon, even though there may be little evidence for their validity.

3.2 BASILISK

My new algorithm, BASILISK, addresses the source of the weaknesses faced by meta bootstrapping. Instead of trusting individual patterns to extract only valid semantic category words, BASILISK uses statistical information from all the patterns that extracted each candidate word. This section describes the various stages of the BASILISK algorithm in detail.

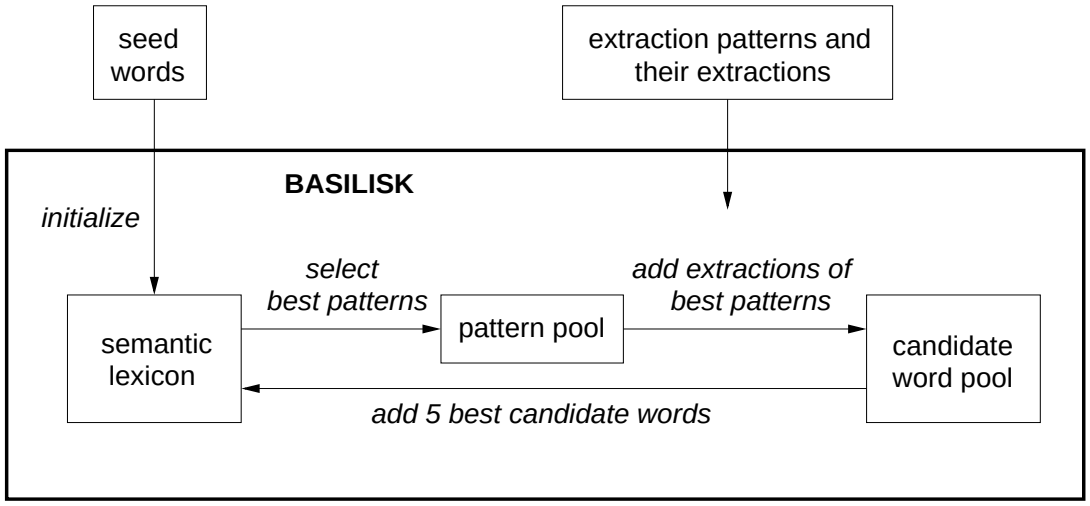

Figure 3.3. BASILISK Algorithm

3.2.1 Seed Word Generation

Like the meta bootstrapping algorithm, BASILISK requires a list of seed words as input. In Riloff & Jones's work, the seed words were chosen by a human, giving total discretion

to the human to select good seed words with no systematic way of doing so. In my research, the algorithm is seeded with the most frequently-occurring head nouns in the corpus. To determine which head nouns occur most frequently, a list of possible seed words is generated by running a partial parser[1] over the corpus and counting the number of times each unique word occurs as the head noun of a noun phrase. A human generates the seed word list for a semantic category by selecting the ten most frequently-occurring unambiguous words belonging to that category.

During this step, BASILISK also uses AutoSlog to generate all extraction patterns from the corpus. The extraction patterns and extractions generated by AutoSlog form the basis of the statistics used by BASILISK throughout its processing.

3.2.2 The Candidate Word Pool

Using a bootstrapping framework similar to mutual bootstrapping, BASILISK begins with a small dictionary of ten seed words and iteratively augments these dictionaries by adding words that are hypothesized to be valid members of each semantic category.

The first step in bootstrapping the semantic lexicon is to find candidate words that might be good words to add. BASILISK first finds the top N extraction patterns as ranked by the $RlogF$ function. All the extractions of those N patterns are added to a candidate word pool from which the best candidate words will be chosen. This step can be seen as a filtering process; instead of choosing from the entire set of head nouns in the corpus, BASILISK narrows the field down to a subset of words that are likely to be good candidates because they are extracted by patterns that are highly associated with the semantic category.

BASILISK uses a value of $N = 20$ patterns[2] at the beginning of bootstrapping, which is sufficient to allow a variety of candidate words to be considered, yet small enough that the extraction patterns are genuinely good patterns for the category. Choosing a larger value of N near the beginning can include many non-category patterns, thus introducing many non-category words into the candidate word pool.

---

[1] I used Sundance, the parser used by the University of Utah NLP research group, for sentence parsing.

[2] "Useless" patterns are not considered. An extraction pattern is useless if all of its extractions already belong to the semantic lexicon. For example, if the pattern "roof of <X>" extracts only two words, and both of them already belong to the *building* lexicon, then that pattern will not be considered one of the best N *building* patterns, because it has no unclassified candidate words to contribute to the candidate word pool.

- Generate all extraction patterns in the corpus and save the patterns with their extractions
- *lexicon* = {seed words}
- $i = 0$
- BOOTSTRAPPING
  1. Score all extraction patterns according to *RlogF*
  2. *pattern_pool* = top ranked 20 + i patterns
  3. *candidate_word_pool* = extractions of patterns in *pattern_pool*
  4. Score candidate words in *candidate_word_pool*
  5. Add top 5 candidate words to *lexicon*
  6. $i := i + 1$
  7. Go to Step 1

Figure 3.4. BASILISK Pseudo-Code

As the algorithm progresses, however, continuing to use a value of $N = 20$ can stifle the candidate word pool. To make this point clear, let us assume that BASILISK performs perfectly, adding only valid category words to the lexicon. After a certain number of iterations, all the valid category members extracted by the top 20 extraction patterns will have been added to the lexicon, leaving only non-category words for the algorithm to consider. For this reason, the pattern pool needs to be infused with new patterns to allow more valid category words to enter the candidate word pool. To achieve this desired effect, BASILISK increments the value of N by one after each iteration of bootstrapping. This action ensures that there will always be at least one new extraction pattern contributing words to the candidate word pool on each successive iteration.

3.2.3 Selection of New Category Words

After creating a pool of candidate words, the next step is to identify the best candidates for the lexicon. One simple scoring metric is the average number of category words extracted by patterns that extracted a candidate word. In this context, a "category word" is a word that has been labeled as belonging to a category in the current semantic lexicon.

The formula for this scoring function is

$$score(word_i) = \frac{\sum_{j=1}^{N_i} F_j}{N_i} \tag{3.3}$$

where N_i is the number of extraction patterns[3] that extract $word_i$, and F_j is the number of unique words in the semantic lexicon that are extracted by pattern j. Intuitively, this scoring function measures the average correlation of a candidate word with a category for each occurrence of the candidate word. It rewards words that co-occur often with category words within many extraction patterns.

Unfortunately, this simple scoring function has a flaw that was discovered during experimentation. In addition to rewarding words that consistently co-occur with category words in a variety of extraction patterns, it also rewards words that co-occur with a large number of category words in a single extraction pattern. For example, let us consider two candidate words, word A and word B. Word A is extracted by ten patterns, each of which extract two category words. Word B is extracted by ten different extraction patterns, one of which extracts twenty category words while the other nine extract none. Word A and word B will receive the same score according to this function. However, word A is more likely to be a valid category word because it consistently co-occurs with known category words. Word B, on the other hand, probably does not belong in the lexicon because there is no systematic correlation between word B and known category words. The only evidence linking word B with the semantic category is a single, high-frequency extraction pattern.

To correct this deficiency in the scoring function, a logarithmic factor is added to lessen the influence of individual, high-frequency patterns on a word's overall score. The improved scoring function used by BASILISK is

$$AvgLog(word_i) = \frac{\sum_{j=1}^{N_i} \log_2(F_j + 1)}{N_i} \tag{3.4}$$

In the previous scoring function, each pattern's contribution to the numerator was the number of category words extracted by that pattern (F_j). In the improved scoring

[3]This formula takes into account the extractions of *every* pattern that extracted $word_i$, not just the patterns that contributed to the candidate word pool.

function, *AvgLog*, each pattern's contribution is $\log_2(F_j + 1)$. The addition of 1 compensates for the fact that if $F_j = 1$, then $\log(F_j) = 0$. This new scoring function will reward words that exhibit a systematic correlation with known category members over the entire body of extraction patterns. Examining its behavior on the previous example, we find that word A (which co-occurred with two category members in each of ten extraction patterns) receives a score of 1.098. In contrast, word B (which only co-occurred with category members in one extraction pattern out of ten) receives a score of 0.300.

As mentioned before, BASILISK tries very hard not to trust any individual pattern too much, preferring to rely on statistics over the entire body of extraction patterns. The exception to this claim occurs when a candidate word is only extracted by a single pattern in the entire corpus. In this case, BASILISK must rely on the evidence provided by a single pattern. If the pattern is a particularly good one, then the candidate word will get a high score and will have a high likelihood of being added to the lexicon. This is not undesirable behavior, for two reasons. First, the absence of negative evidence can be interpreted as weak positive evidence. Since the candidate word is extracted by no other patterns except a single good one, it is likely that it may be a valid semantic category member. During experimentation with BASILISK, visual inspection of these instances confirmed that this is often the case. Second, even if the word is not truly a member of the semantic category, its negative effect on the rest of the bootstrapping process will be small because it is only extracted by a single pattern.

Once the words in the candidate word pool are scored, they are ranked and the top five are added to the semantic lexicon. The pattern pool and candidate word pool are cleared, and the process repeats from the beginning.

3.3 Single Category Results

This section compares the results of meta bootstrapping to the results of BASILISK. In these experiments and all other experiments presented in this thesis, I used the MUC-4 terrorism corpus [1], which contains 1700 newswire articles about terrorism in Latin America. Performance on six categories will be explored in this thesis: *building*, *event*, *human*, *location*, *time*, and *weapon*.

Results for both algorithms can be seen in Figure 3.5. Some things should be noted when examining the graphs presented in this thesis. First, tables showing some of the data points represented by these graphs can be found in Appendix B. Second, the vertical

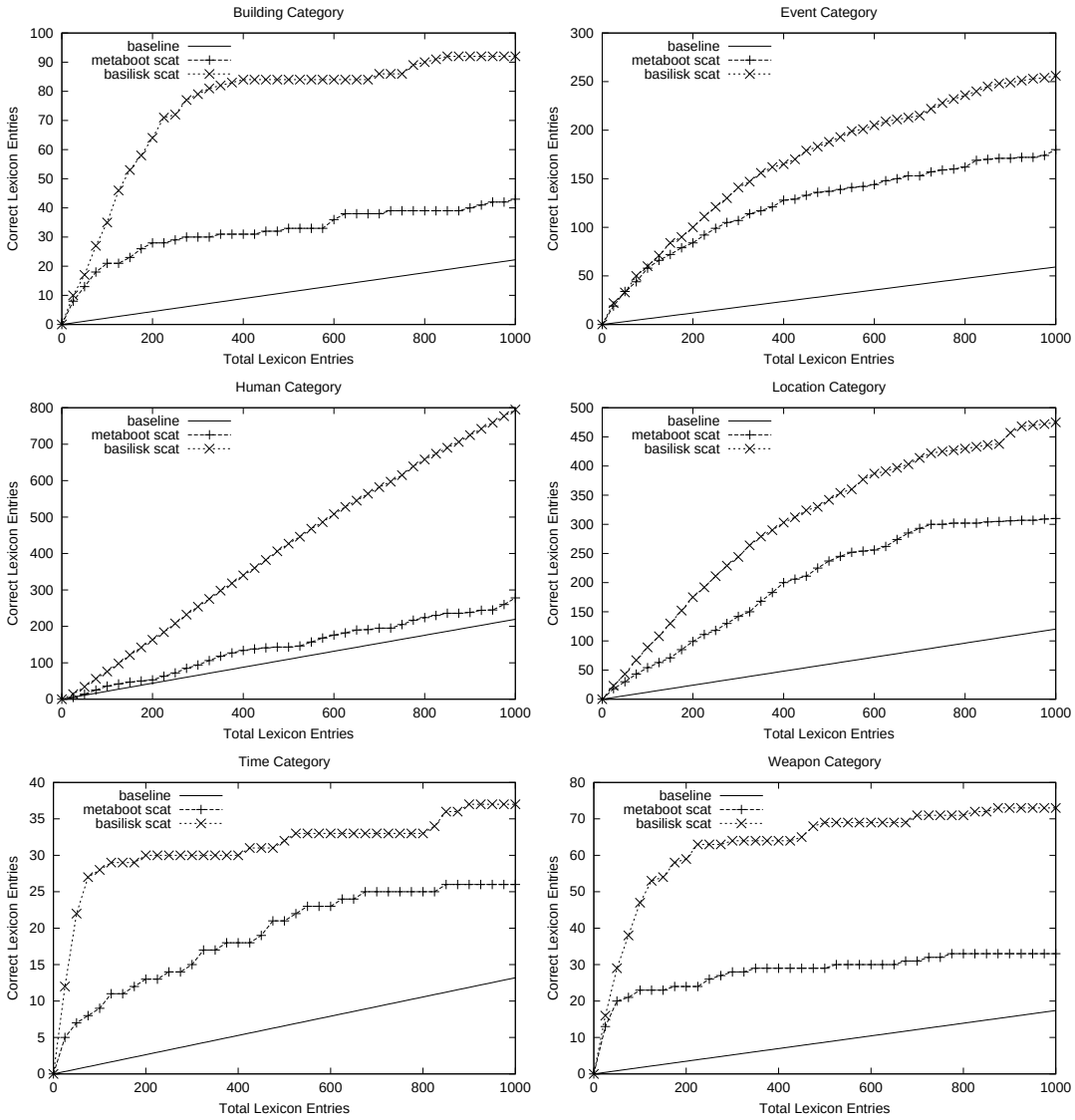

Figure 3.5. Meta Bootstrapping vs. BASILISK, Single Category

axes in the graphs are not all the same. Because different categories yield varying degrees of success, it is difficult to graph various algorithms' results for different categories while maintaining fixed axes. For this reason, it is important to note that the vertical axes on various graphs may be different. Third, the "baseline" represents the performance that could be expected if words were chosen at random. It is simply a line representing the conditional probability that a random word belongs to a category.

The meta bootstrapping results are not comparable to the results published in the orig-

inal meta bootstrapping paper [19], because Riloff & Jones generated semantic lexicons of full noun phrases, while I used meta bootstrapping to generate semantic lexicons of head nouns only. In general, the results using only head nouns are lower than the results using full noun phrases. This occurs because unique phrases that share a head noun will only be scored once when scoring head nouns, whereas they may have been rewarded separately when scoring full noun phrases. For example, Riloff & Jones would have scored "eastern Colombia" and "Colombia" separately, but BASILISK does not distinguish between the two because they have the same head noun, "Colombia".

BASILISK clearly outperforms meta bootstrapping for every category. The major difference between meta bootstrapping and BASILISK is the way in which extraction patterns are used to hypothesize words for the semantic lexicon. Meta bootstrapping is pattern-centric while BASILISK's approach is word-centric. Since BASILISK outperforms meta bootstrapping in all cases presented, there is strong evidence for the first claim of this thesis: that utilizing collective statistics over a body of extraction patterns can improve the precision of automatically-generated semantic lexicons.

CHAPTER 4

ALGORITHMS FOR MULTIPLE CATEGORIES

A major claim of this thesis is that generating semantic lexicons for multiple categories simultaneously will increase the overall precision of the lexicons. Why would generating semantic lexicons for multiple categories simultaneously be expected to improve performance? My hypothesis was that errors of confusion[1] between semantic categories can be lessened by having information about all categories available.

Many extraction patterns exhibit a systematic ambiguity between two or more categories. As we saw in Figure 1.1, the pattern "was killed in <X>" can extract many valid *location* words, but can also extract many *event* words such as "shootout" and "clashes". When generating semantic lexicons for a single category, such ambiguity can cause the lexicon to become infected with words that really belong to another semantic category. For example, when generating only a *location* lexicon, the algorithm may choose "shootout" because it has no idea that the pattern tends to extract both *location* and *event* words. It can't know this because it does not even know that the *event* category exists. However, when generating lexicons for multiple categories simultaneously, the algorithm can detect such ambiguity and handle the situation appropriately. With this ability, errors of confusion between *location* and *event* words can be reduced.

Figure 4.1 shows another way of viewing the task of semantic lexicon generation. In this figure, the set of all words in the corpus is visualized as a search space. Each category owns a certain territory within the search space (demarcated with a dashed line), representing the words that are true members of that category. Not all territories are the same size, since some categories have more words than others. Some categories have bordering territory, representing cases where there may be systematic ambiguity of extraction patterns that extract members of both classes. Of course, this visualization

[1]Throughout this thesis, I use the term *confusion* to refer to errors where a word belonging to category X was incorrectly added to the lexicon for category Y.

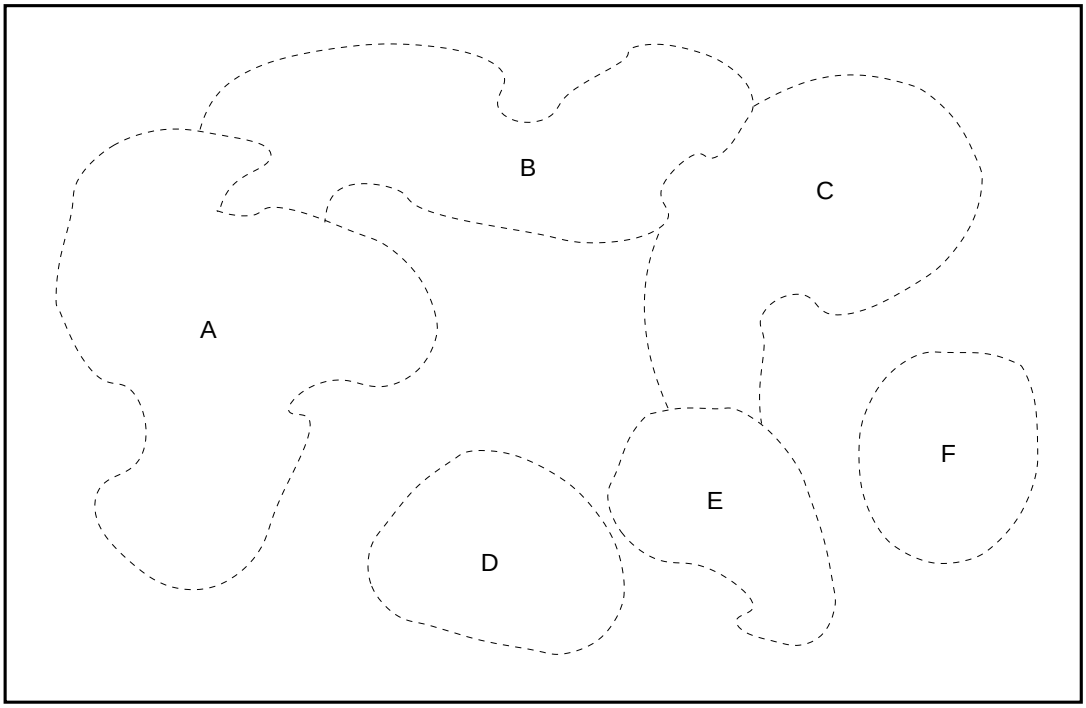

Figure 4.1. Visualization of Search Space

is a simplification of the true search space, but it serves well to illustrate the concepts presented in this chapter.

Figure 4.2 demonstrates what happens when a semantic lexicon is generated for a single category. The seed words for the category (in this case, category C) are represented by solid black areas within the category's territory. The hypothesized words in the growing lexicon are represented by a shaded area. The goal of the lexicon generation algorithm is to expand the area of claimed words so that it exactly matches the category's true territory in the search space. If the claimed area expands beyond its true territory, then the category's lexicon includes incorrect words. In this example, category C has claimed a significant number of words beyond its own territory.

Much of the territory claimed by category C actually belongs to other categories, resulting in confusion errors. When generating a lexicon for a single category at a time, such errors are impossible to detect. In this case, the algorithm only knows about category C and does not know about territory belonging to other categories. As a consequence, category C is free to expand in any direction, even deep into the heart of territory

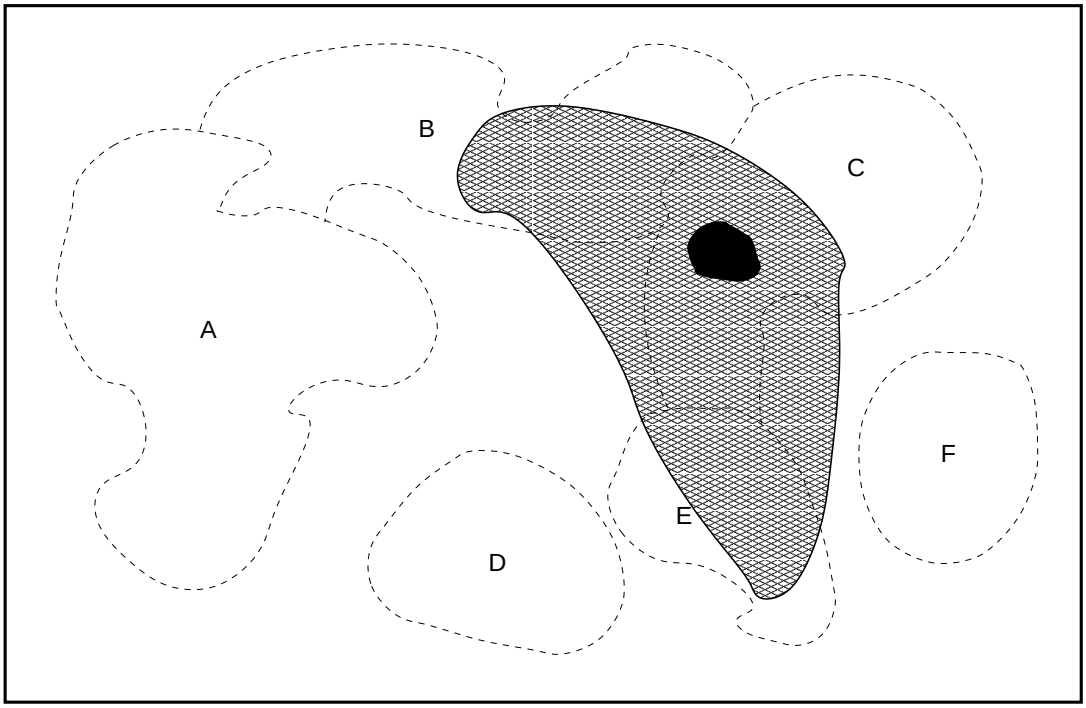

Figure 4.2. Bootstrapping a Single Category

belonging to categories B and E.

However, while generating lexicons for multiple categories, simple conflict resolution disallows each semantic category from claiming a portion of the search space that has already been claimed by another category. Since the lexicons are not allowed to overlap, the ability of categories to overstep their boundaries is limited. Confusion errors may still occur; however, their scope is more limited because knowledge about all categories is available.

Figure 4.3 shows the same search space when lexicons are generated for all categories simultaneously. When generating lexicons only for one category, category C was able to claim extensive territory belonging to categories B and E. However, when generating lexicons for all categories simultaneously, category C is limited in its ability to expand into other categories' territory. Because claims of different categories are not allowed to overlap, category C is forced to expand in other directions once it encounters territory claimed by B and E. In this example, category C is benefitted by being forced to expand further into its own territory.

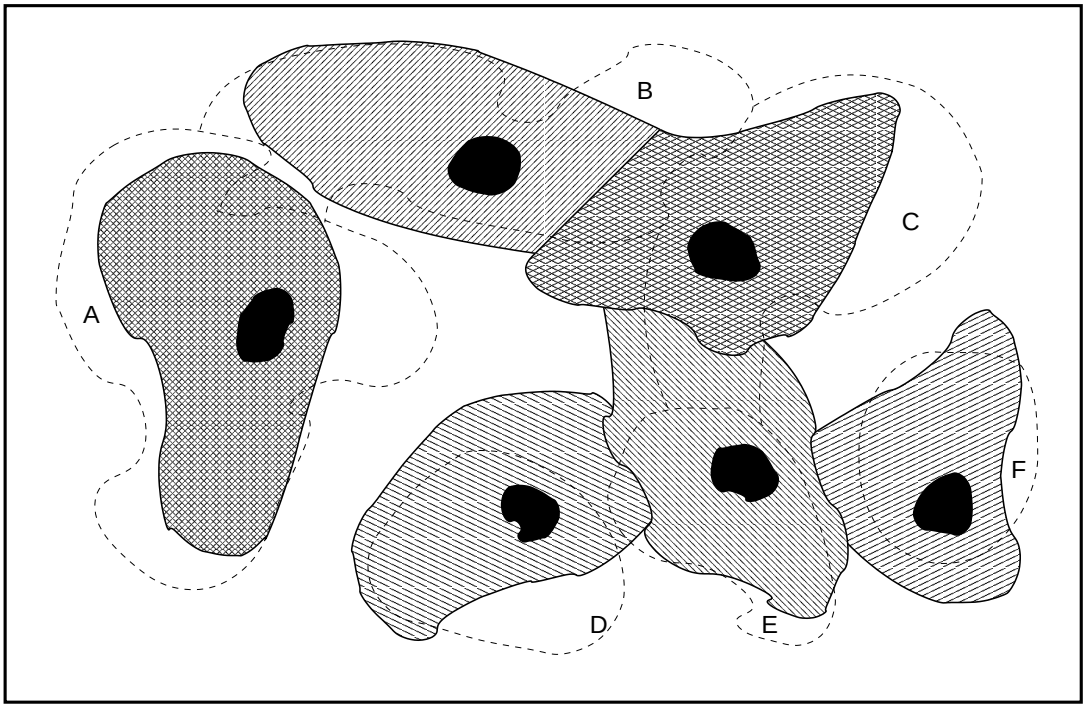

Figure 4.3. Bootstrapping Multiple Categories Simultaneously

Section 4.1 describes how meta bootstrapping and BASILISK can be extended to use simple conflict resolution in generating lexicons for multiple categories. Section 4.2 also describes how BASILISK can be extended to take further advantage of multiple categories simultaneously by using a more intelligent scoring function to choose words for the lexicon. In the comparison of the algorithms and their performance, the abbreviation SCAT represents a "single category at a time", while the abbreviation MCAT represents "multiple categories simultaneously."

4.1 Simple Conflict Resolution

Perhaps the easiest way to take advantage of multiple semantic categories simultaneously is to add simple conflict resolution into the algorithm to enforce the "one sense per domain" assumption. Whenever a word is hypothesized to belong to more than one category, the algorithm must choose one, since a word can only be added to the lexicon for one category.

Simple conflict resolution can be added to the original meta bootstrapping and BASILISK

algorithms in a fairly straightforward manner, allowing them to take advantage of multiple categories simultaneously. The next two sections describe how simple conflict resolution can be integrated into the meta bootstrapping and BASILISK algorithms, and how the precision of the final semantic lexicons is improved as a result.

4.1.1 Meta Bootstrapping

The meta bootstrapping MCAT algorithm using simple conflict resolution is extremely similar to the original meta bootstrapping algorithm. In fact, the only modification that needs to be made is to perform the conflict resolution step during mutual bootstrapping when words are added to the temporary lexicon.

Each iteration of mutual bootstrapping is done in parallel for the various categories, ensuring that no word can enter the temporary lexicon for more than one category. During each iteration, the conflict resolution is performed after the best pattern for each category is chosen. At that point, if a word is extracted by the best pattern for more than one category, the word is scored using the word-scoring function (Equation 3.2). The conflict is resolved in favor of the category for which the word receives a higher score, and the word is added to the temporary lexicon for that category.

There is no modification to be done to the meta bootstrapping layer to implement this conflict resolution. Since a word can only be added to one category during mutual bootstrapping, there can be no conflicts in the temporary lexicon when the meta bootstrapping layer chooses words for the permanent lexicon.

Examining the results shown in Figure 4.4, one can see that every category except *time* benefits from generating multiple categories simultaneously. In fact, using multiple categories simultaneously improves overall performance for all categories (except *time*) by an approximate factor of two.

4.1.2 BASILISK

Adapting BASILISK to handle multiple categories using simple conflict resolution also requires minimal changes to the algorithm. The conflict resolution must be performed after the candidate words are scored, as BASILISK attempts to add them to the lexicon. If any word is hypothesized for more than one category, then it is scored for those categories and added to the category for which it receives the highest score. This conflict resolution is the only significant change to the algorithm.

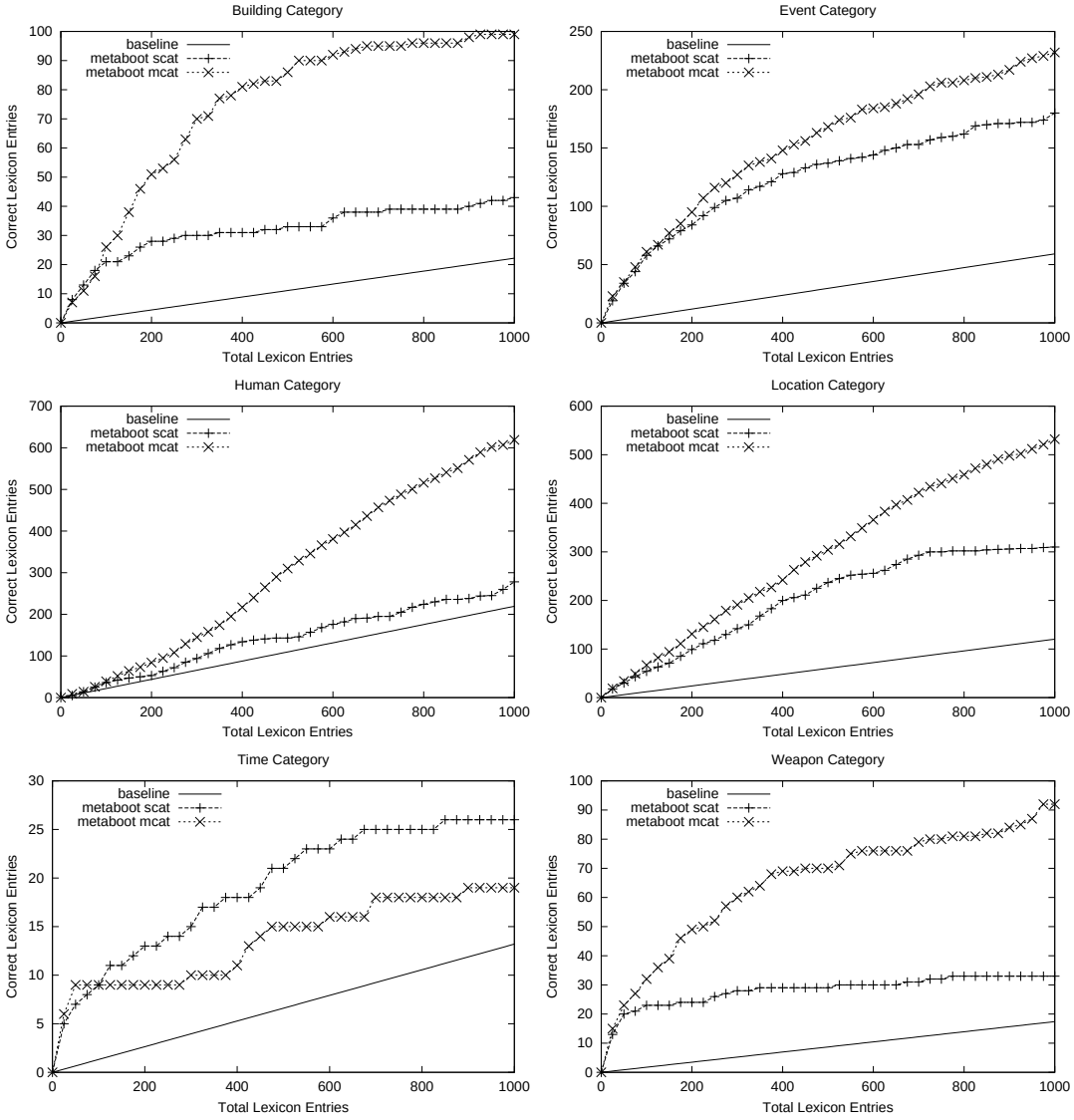

Figure 4.4. Meta Bootstrapping SCAT vs. Meta Bootstrapping MCAT

The results in Figure 4.5 reveal that using simple conflict resolution in generating lexicons for multiple categories simultaneously improves the performance of BASILISK a small amount overall. The *weapon* and *building* categories benefit most, and the improvement in all cases occurs after several iterations of bootstrapping rather than at the beginning. In fact, for the *event*, *human*, and *location* categories, the difference is hardly perceptible until 600 words have been generated. This phenomenon is consistent with the visualization of the search space in Figure 4.3. Since the seed words of each category are not generally located near each other in the search space, the algorithm is relatively unaffected in the earlier bootstrapping iterations. The conflict resolution only takes effect once the categories begin encroaching on each other's territory.

It can be seen in Figure 4.5 that not all categories generate a full 1000 words when BASILISK MCAT is run for 200 iterations. For example, the *building* category only generates 706 words. Each category is expected to generate 1000 words during 200 iterations because five words are chosen during each iteration. Why might fewer words be generated? This may happen when the simple conflict resolution resolves many disputed words in favor of other categories. Although the *building* category is allowed to choose five words to add to the lexicon, disputed words are resolved in favor of the category for which each word receives a higher score. If there are fewer than five words that receive a higher score for the *building* category than for another category, then fewer than five words will be added to the *building* category. This effect is rare in the early stages of bootstrapping. However, in later iterations conflicts are abundant, and some categories are often left with fewer than five words at each iteration.

As one can see from the results, this effect may actually be beneficial to the category losing the disputed words. For example, the *building*, *location*, and *weapon* categories all generate fewer than 1000 words. However, even when generating fewer words, the actual number of correct lexicon entries for each category is higher for MCAT than SCAT. The precision of the lexicons is increased by correctly disambiguating disputed words in favor of a category for which there is stronger evidence. The results indicate that this is true even when conflict resolution leaves fewer than five words for the category losing the dispute.

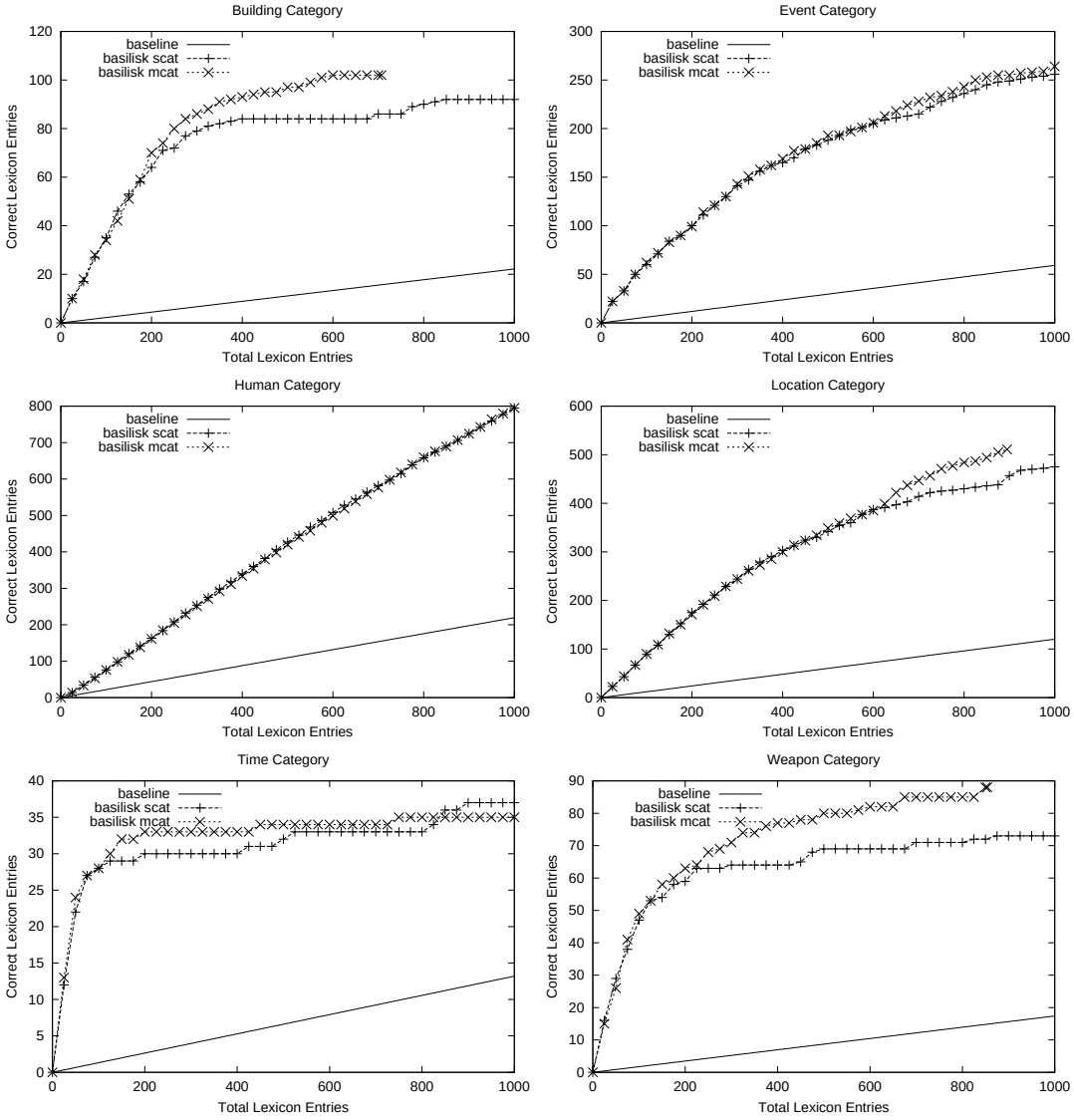

Figure 4.5. BASILISK SCAT VS. BASILISK MCAT

4.2 A Better Scoring Function

Although adding a simple method of conflict resolution improves the precision of BASILISK's lexicons, it is not a particularly intelligent way to take advantage of multiple categories simultaneously. It is preferable to use an algorithm that makes active use of the information available from all categories, rather than an algorithm that passively enforces a simple constraint. In this way, the search space may be more intelligently explored because each category "knows" where the other categories are located. Using only simple conflict resolution, each category explores the search space until it directly encounters another category's claimed territory, into which it cannot expand. A smarter algorithm will try to avoid the other categories' territory before such a direct encounter, choosing instead to explore areas of the search space that do not belong to any category. In terms specific to BASILISK, the algorithm will prefer words that co-occur frequently with other words from the category under consideration, and do not co-occur frequently with words from other categories.

BASILISK MCAT+ is an enhancement of BASILISK MCAT that takes advantage of multiple categories in such a manner. BASILISK MCAT+ uses a variation of BASILISK's candidate-scoring function to choose words that have strong evidence for belonging to one category but no evidence or only weak evidence for belonging to others. When scoring a candidate word for a category, the word is penalized according to its maximum score for any other category. Each word in the candidate word pool for a category receives a score according to the following formula:

$$score(word_i, cat_a) = AvgLog(word_i, cat_a) - \max_{a \neq b}(AvgLog(word_i, cat_b)) \qquad (4.1)$$

where *AvgLog* is the candidate-scoring function used by BASILISK SCAT (Equation 3.4). This new scoring metric takes advantage of the information available from multiple categories by penalizing words that receive high scores for more than one category. For example, if BASILISK MCAT+ is scoring candidate words for the *location* category, then each word's score will be its score for *location* minus its maximum score for any other category. A word that receives a high *location* score according to the original scoring function will only receive a high overall *location* score for BASILISK MCAT+ if its scores for other categories are low.

This allows the search space to be more intelligently explored, as we can see by visualizing the problem as in Figure 4.3. We have seen that BASILISK MCAT is exactly the same as BASILISK SCAT except for the addition of a simple conflict resolution scheme.

Each category will explore the search space until it stumbles across territory claimed by another category, at which point it is forced to explore in other directions. In contrast, during BASILISK MCAT+, each category "knows" where the other categories are because information from other categories is explicitly used when scoring candidate words in Equation 4.1. In this way, each category will prefer to explore new areas of the search space while steering clear of the territory claimed by other categories.

Figure 4.6 compares the MCAT algorithms described in this chapter: meta bootstrapping MCAT, BASILISK MCAT, and BASILISK MCAT+. BASILISK MCAT+ generally performs better than BASILISK MCAT, although the improvement is small for most categories. For the *event* category, BASILISK MCAT actually outperforms BASILISK MCAT+. In all cases, both versions of BASILISK perform with higher precision than meta bootstrapping, which supports the claim that collective evidence over many extraction patterns is helpful, even when generating lexicons for multiple categories simultaneously. Overall, the improved scoring function is a very small win over MCAT with simple conflict resolution.

This chapter has presented evidence that generating semantic lexicons for multiple categories simultaneously can improve the precision of the lexicons. Even adding a very simple conflict resolution scheme can provide great benefits, particularly for meta bootstrapping. BASILISK MCAT and BASILISK MCAT+ take advantage of both major claims of this thesis: that precision of semantic lexicons can be improved by utilizing collective evidence over many extraction patterns and by generating lexicons for multiple categories simultaneously. The performance of those algorithms provides evidence to support both claims.

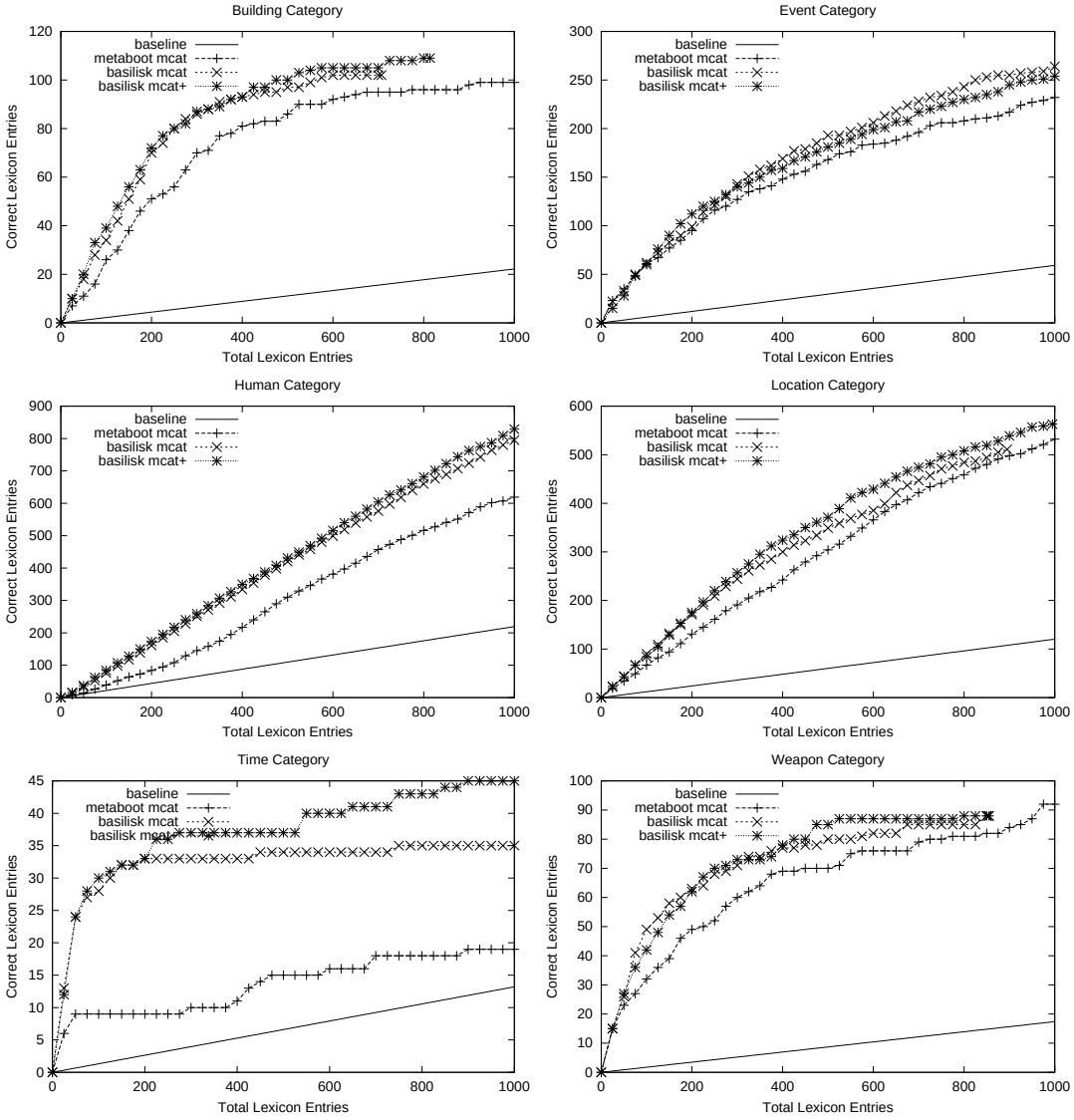

Figure 4.6. Meta Bootstrapping MCAT vs. BASILISK MCAT vs. BASILISK MCAT+

CHAPTER 5

ANALYSIS AND CONCLUSIONS

This chapter analyzes the data presented in this thesis, describes benefits and limitations of BASILISK, provides suggestions for future research, and offers conclusions.

5.1 Benefits of BASILISK

5.1.1 Higher Precision

Semantic lexicons generated by BASILISK generally have higher precision than those generated using previous techniques. To see the improvement in precision made by BASILISK over previous techniques, Figure 5.1 compares all algorithms presented in this thesis. As one would hope, all the algorithms perform above the random baseline. In general, meta bootstrapping SCAT shows the weakest performance, with meta bootstrapping MCAT and BASILISK SCAT providing large improvements. BASILISK MCAT generally outperforms both of these algorithms, with BASILISK MCAT+ outperforming everything on nearly all categories.

There are many clear benefits of having automatically-generated, domain-specific semantic lexicons with high precision. Generating semantic lexicons automatically saves much of the time and human effort of compiling lexicons by hand. It also helps eliminate human errors of omission, particularly with domain-specific language that may not be known to the person compiling a lexicon.

Although generating domain-specific lexicons automatically saves some human effort, some effort will need to be expended to filter the lexicons. The relatively high precision of the lexicons generated by BASILISK alleviates some of this effort. In particular, filtering a lexicon with low precision can be a boring, frustrating task. Since BASILISK's higher-precision lexicons include many valid category words near the beginning of each lexicon, the filtering is less tedious, more fruitful, and moves more quickly. Also, if only a certain number of category words are needed, the human will not have to scan as many words if the lexicon has high precision.

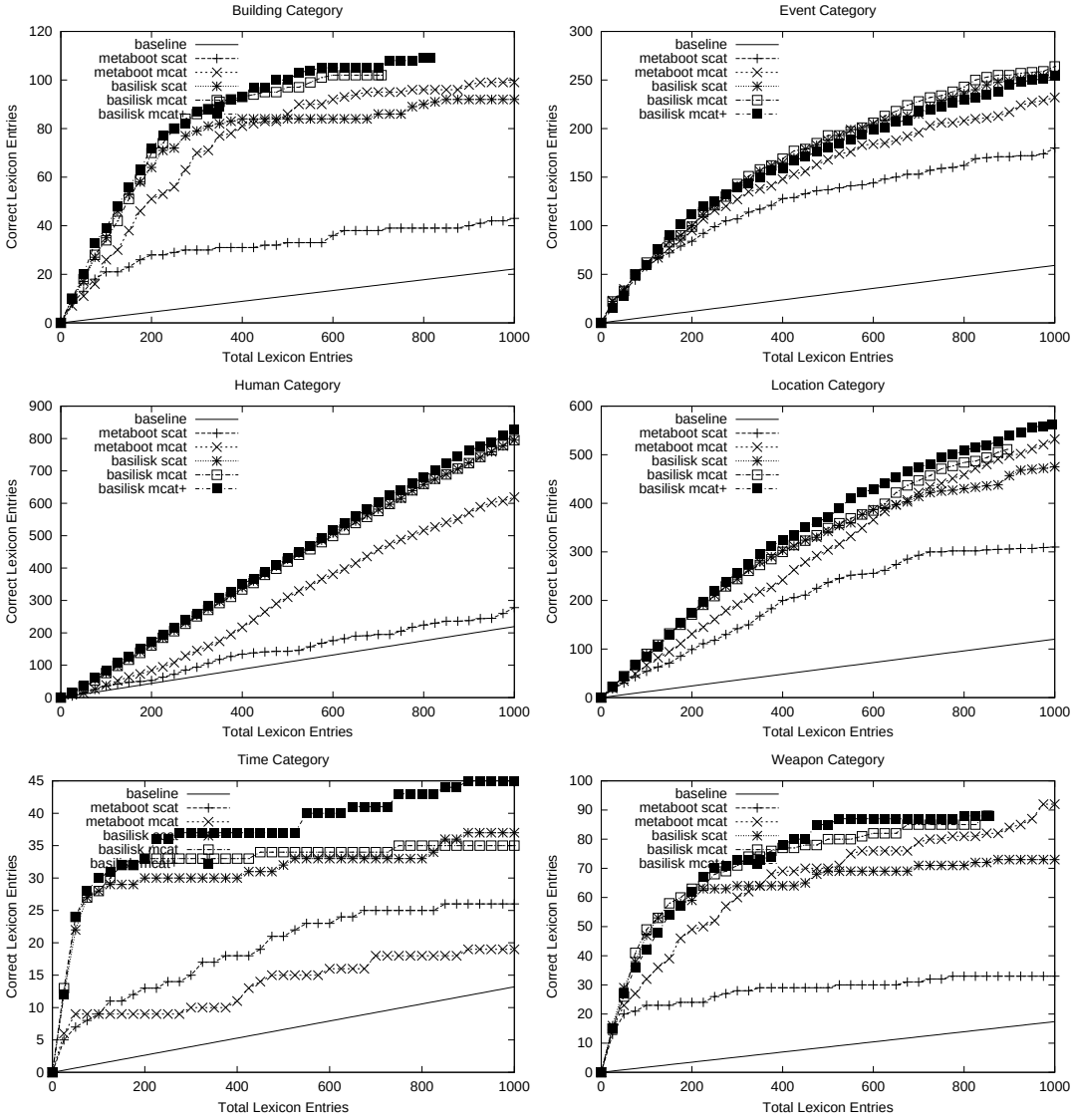

Figure 5.1. Performance of All Algorithms

The top 50 words for each category in the final lexicon generated by BASILISK MCAT+ are shown in Figure 5.2. One can see that there are many valid lexicon entries for most categories. Still, there is much improvement to be made, particularly for the less common categories such as *time* and *building*.

5.1.2 Higher Recall

In addition to improved precision, BASILISK also achieves greater recall than previous techniques. Whereas precision measures the percentage of correct words in the lexicon, recall measures the percentage of all true category members found by the algorithm. The problem of achieving high recall is the problem of finding every category word in the corpus, thus eliminating errors of omission. This is one of the main problems of the semantic lexicon generation task, and one of the main benefits of generating lexicons automatically.

A table of recall results can be found in Appendix C. When it is allowed to run for 200 iterations, BASILISK achieves recall in the 40–60% range, which is very respectable. Comparison to previously published results is not possible, because in previous work recall could not be measured. Recall for this task can only be directly measured if one knows the true number of words in the corpus that belong to each category. In the course of experimentation, every word has been examined and given a category label, so these data are known.

5.1.3 Reduction of Confusion Errors

Generating lexicons for multiple categories simultaneously reduces the confusion between categories. In other words, although BASILISK generates many incorrect lexicon entries, few of those entries belong to other categories. Most of the incorrect lexicon entries do not belong to any category. This effect is to be expected if BASILISK MCAT+ is taking advantage of multiple categories as it should. The reduction of confusion between categories is further evidence that utilizing multiple categories simultaneously has the desired effect.

Tables of confusion data are presented in Appendix D. Results for all algorithms are compared. Generating multiple categories simultaneously provides the greatest confusion reduction, and for some categories, confusion can be reduced by over 80%.

Building: *ladrillo cela restroom theatre hilt promenade cai store checkpoint ciclotecnica esquipulas manzanillo gas_station cathedral crossroads door strongholds premises flames obaldia 0ll5 temple marker doors palace reopening spaces entrance borders penitentiary km-28 disposition bank_branch disposal gmt terminal present location branches buildings intolerance sanctuary academy briefing facilities houses school police_station park mansions*

Event: *exercises disobedience exploitation tragedies ambush disturbance act accident aggression assassination 6-hour violating uprisings producing manipulations liquidation authorship sabotage dynamite_attacks bomb_attack countless dynamite_attack annihilation harassment bomb_attacks detonating mafiosi attemp sabotages placement burning planting placing barletta murders games destruction takeover executions maneuvers incursion kidnappings intervention clash uprising shoot-out bomb_explosion interference controntation releases*

Human: *boys maoist_people's_liberation_army eln_camilist_union ernesto snipers french unmasking national_guard permission individuals detainees traveling army_of_national_liberation pizarro the_extraditables leader commandoes politicians diaz professionals extremists activists foreigners espinosa women popular_liberation_army cordero persons gacha deserter riding individual narcoterrorists demonstrators policemen cronies celades meneses brings missionaries citizen woman cazolo priests escobar policeman cano leftists civilian rivera*

Location: *ruins valparaiso san_salvador soyapango mejicanos vitoria oslo regions marta corregimiento usulutan zacatecoluca credisa san_miguel suburb interior quito capital tonacatepeque delgado quilcura jabaquara platinedi huaraz spots halls isidro maria tegucigalpa cities lhasa neighborhoods departments complying region totonique salvado achi cycles chalatenango morena ayutuxtepeque cuscatancingo gotera alegria copinol chirilagua majatepeque unicentro zacateluca*

Time: *owing ride showing disruptions top afternoon km starting evening decade somewhere blocks hour march weeks sabotaging 1/2 saturday december january spot plains july tuesday friday april february october june september flatlands thurday thursday's monday august atlantic suspicion sunday floor growth eve wednesday antequera motorcycles happiness battlefield coast bail telecommunications anniversary*

Weapon: *medication yearning lumber belongings machines sticks minerals cannon stones airliner grenade casings dynamite_charge launchers materials firebomb molotov_cocktail dynamite_sticks car-bomb rifle truck_bomb medium-intensity carbomb pistol fuses cannons kg car_bombs car_bomb package 20-kg explosive_charge incendiary load device callao banner ardito dynamite_charges machineguns circle weapon processing roadblocks cocaine good-bye fruit firearms ammunition music*

Figure 5.2. Final Lexicons Generated by BASILISK MCAT+

5.1.4 Efficiency

In addition to the improvement in recall and precision of the final lexicons, BASILISK provides an efficiency gain over meta bootstrapping. It is fair to compare one iteration of meta bootstrapping to one iteration of BASILISK, since they both add five words to the lexicon. Each iteration of meta bootstrapping consists of ten iterations of mutual bootstrapping, during which every extraction pattern must be scored to find the best one. Thus, for each iteration of meta bootstrapping, all extraction patterns must be scored ten times. In contrast, each iteration of BASILISK only needs to score every extraction pattern once. For this reason, BASILISK runs approximately ten times as quickly as meta bootstrapping. With 40,967 extraction patterns in the corpus, the time savings is significant.

5.2 Limitations of BASILISK

Although BASILISK achieves higher precision than previous techniques, it has some limitations that need to be addressed.

5.2.1 Stopping Conditions

First, BASILISK lacks a clear stopping condition in SCAT and MCAT modes. The algorithm can be run for as many iterations as the user likes, stopping after generating a certain number of words. Alternatively, a threshold can be used, stopping the algorithm when word scores drop below a certain level. Currently, however, there is no sophisticated way of letting BASILISK know that its performance has dropped below an acceptable level or that it has found all the category members that can reasonably be found.

On the other hand, BASILISK MCAT+ can be given a simple stopping condition that works fairly well. The algorithm can refuse to add any word to the lexicon for a category if it receives a negative score for that category. When a word receives a negative score for a category, that indicates that there is greater evidence that it belongs to a different category. Enforcing this constraint allows BASILISK to stop bootstrapping a category when all remaining words have greater evidence of belonging to a different category.

When applying this simple constraint, BASILISK MCAT+ generates the following number of words for each category:

- *building*: 359 total, 93 correct, 25.9% precision
- *event*: 1000 total, 259 correct, 25.9% precision

- *human*: 1000 total, 831 correct, 83.1% precision
- *location*: 1000 total, 492 correct, 49.2% precision
- *time*: 159 total, 35 correct, 22.0% precision
- *weapon*: 333 total, 78 correct, 23.4% precision

These numbers are neither significantly better nor significantly worse than BASILISK MCAT+'s performance without the stopping condition (except for *location*, which does worse with the stopping condition). Whether the algorithm stopped at the correct time for each category is a subjective decision involving a trade-off between recall and precision. As more words are added to the lexicon, its precision generally gets worse, but its recall can only improve. Often, it is necessary to make a decision about whether recall or precision is more important. If recall is more important then precision can be sacrificed, and vice versa. A good stopping condition is not always necessary if recall is more important; however, if precision is preferred then a good stopping condition is crucial. For this reason, developing a better stopping criterion for BASILISK may be very important.

5.2.2 Mutual Exclusivity Assumption

One assumption of this thesis is the "one sense per domain" assumption, that a word is only used in a single sense within a limited domain. This assumption of mutual exclusivity is true most of the time, and it exemplifies the benefit of domain-specific lexicons as explained in Chapter 1. However, even within a limited domain, there are significant violations of this assumption. For example, certain words can often refer to either people or locations. An example of this usage in the MUC-4 terrorism corpus is "Flores", the name of a Guatemalan political leader as well as the name of a city in Guatemala.

Ambiguity such as this can cause problems in two ways. First, since the word "Flores" can refer to either a *human* or *location*, it can often co-occur with both *human* and *location* words in extraction patterns. This can create confusion between the *human* and *location* categories during bootstrapping. Second, ambiguous words present a problem when scoring the final lexicons. Should "Flores" be scored as a *human* or a *location*? For results presented in this thesis, all occurrences of each ambiguous word were found, and the word was scored according to its most frequent usage. In this case, "Flores" was scored as *human* because it most frequently refers to the Guatemalan leader Gilda Flores.

5.2.3 Poorly Represented Categories

Another limitation of BASILISK is that it can have difficulty finding valid words for categories that are poorly represented in the corpus. For example, BASILISK performs poorly on the *time* category, for which there are only 112 unique words in the texts. BASILISK's performance is better than previous techniques, but even at its best, BASILISK only achieves precision of about 30% for the *time* category. Since BASILISK relies heavily on co-occurrence of words within many extraction patterns, sparse data can cause its performance to suffer.

5.2.4 Correlation Between Categories

BASILISK's performance can also suffer when there is a systematic correlation between categories such that members of each category tend to occur within the same extraction patterns. This phenomenon can be demonstrated with the *vehicle* category. Originally, *vehicle* was one of the categories I used during experimentation. It is the least represented category in the corpus, with only 89 unique words. When running in MCAT mode with the *vehicle* category, BASILISK's performance on the *building* category decreased noticeably. Upon further investigation, I discovered that many *building* words had been stolen by the *vehicle* category early in the bootstrapping process. More investigation showed that this was occurring because both *building* and *vehicle* words are often extracted by patterns having to do with attacks against a target, such as "destroyed <X>". In the MUC-4 terrorism corpus, both buildings and vehicles are often damaged in attacks, and so there is a systematic correlation between the two categories that was not obvious at first. Although buildings and vehicles are clearly separate semantic categories in a general sense, they end up as submembers of a conceptual *target* category in the terrorism domain. Normally, BASILISK is very good at harnessing the mutual exclusivity of semantic categories to prevent systematic confusion such as this. However, in this case, *vehicle* was able to steal words from the *building* category because of the systematic correlation between the two categories and the relative scarcity of both. Because neither of the categories is well represented in the corpus, BASILISK was unable to take advantage of its major strength, the collective evidence over many extraction patterns. For this reason, the categories to be used must be carefully chosen by the user. Generally, BASILISK performs best with categories that are well represented and not correlated in any systematic way.

5.3 Future Work

Future research in this area should address the limitations of BASILISK, such as its sensitivity to category selection and its poor performance in low-frequency categories. Research may also be devoted to finding a better method for harnessing multiple categories simultaneously. Although the methods presented in this thesis perform fairly well, the improvements provided by BASILISK MCAT+ are small and somewhat inconsistent. Developing a method to take greater advantage of multiple categories should be a fruitful area for future research. Also, it would be a useful experiment to port BASILISK to a different domain than the MUC-4 terrorism newswire articles, to prove its generality.

Although BASILISK focuses on generating semantic lexicons with high precision, it may be useful to explore opportunities to generate extraction patterns as well. Extraction patterns themselves are especially useful for information extraction. The most straightforward way of generating extraction patterns might be to score and rank them according to the *RlogF* function and the contents of BASILISK's final lexicons. However, adapting BASILISK to produce extraction patterns more intelligently may be a worthwhile pursuit.

5.4 Conclusions

This thesis claims that precision of automatically-generated semantic lexicons can be improved by utilizing collective evidence over a body of extraction patterns and by generating lexicons for multiple categories simultaneously.

Chapter 3 introduced a new algorithm, BASILISK, that uses statistics from many extraction patterns to choose words for a semantic lexicon. The results from that section provide strong evidence to substantiate the first claim.

Chapter 4 introduced methods for extending BASILISK and meta bootstrapping to generate lexicons for multiple categories simultaneously. The results from that section provide evidence to substantiate the second claim. Significantly, it was shown that taking advantage of both claims together can provide a greater benefit than using either one alone.

APPENDIX

SEED WORDS

Figure A.1 shows the seed words used for each category for all algorithms described in this thesis. The ten most frequently-occurring words for each category were chosen as seed words.

Building: *embassy office headquarters church offices house home residence hospital airport*
Event: *attack actions war meeting elections murder attacks action struggle agreement*
Human: *people guerrillas members troops cristiani rebels president terrorists soldiers leaders*
Location: *country el_salvador salvador united_states area colombia city countries department nicaragua*
Time: *time years days november hours night morning week year day*
Weapon: *weapons bomb bombs explosives arms missiles dynamite rifles materiel bullets*

Figure A.1. Seed Words

APPENDIX

PRECISION RESULTS

Table B.1 and Table B.2 present precision data for all algorithms described in this thesis. The "Total" column represents the number of words generated, in increments of 100. Underneath the name of each algorithm are two numbers. The first number is the number of correct lexicon entries of the first N words hypothesized by that algorithm, where N is the number in the "Total" column. The second number, in parentheses, shows the corresponding precision of the lexicon (i.e., the first column divided by the "Total" column).

building					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	21 (21.0%)	35 (35.0%)	26 (26.0%)	34 (34.0%)	39 (39.0%)
200	28 (14.0%)	64 (32.0%)	51 (25.5%)	70 (35.0%)	72 (36.0%)
300	30 (10.0%)	79 (26.3%)	70 (23.3%)	86 (28.7%)	87 (29.0%)
400	31 (7.7%)	84 (21.0%)	81 (20.3%)	93 (23.3%)	93 (23.3%)
500	33 (6.6%)	84 (16.8%)	86 (17.2%)	97 (19.4%)	100 (20.0%)
600	36 (6.0%)	84 (14.0%)	92 (15.3%)	102 (17.0%)	105 (17.5%)
700	38 (5.4%)	86 (12.3%)	95 (13.6%)	102 (14.6%)	105 (15.0%)
800	39 (4.9%)	90 (11.3%)	96 (12.0%)		109 (13.6%)
900	40 (4.4%)	92 (10.2%)	98 (10.9%)		
1000	43 (4.3%)	92 (9.2%)	99 (9.9%)		
event					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	61 (61.0%)	61 (61.0%)	64 (64.0%)	64 (64.0%)	61 (61.0%)
200	89 (44.5%)	103 (51.5%)	102 (51.0%)	102 (51.0%)	114 (57.0%)
300	113 (37.7%)	145 (48.3%)	138 (46.0%)	147 (49.0%)	143 (47.7%)
400	137 (34.3%)	173 (43.2%)	158 (39.5%)	176 (44.0%)	163 (40.7%)
500	146 (29.2%)	196 (39.2%)	179 (35.8%)	200 (40.0%)	186 (37.2%)
600	153 (25.5%)	214 (35.7%)	197 (32.8%)	214 (35.7%)	208 (34.7%)
700	162 (23.1%)	226 (32.3%)	209 (29.9%)	238 (34.0%)	227 (32.4%)
800	172 (21.5%)	247 (30.9%)	221 (27.6%)	253 (31.6%)	240 (30.0%)
900	181 (20.1%)	261 (29.0%)	230 (25.6%)	266 (29.6%)	256 (28.4%)
1000	190 (19.0%)	272 (27.2%)	245 (24.5%)	276 (27.6%)	266 (26.6%)
human					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	36 (36.0%)	76 (76.0%)	39 (39.0%)	76 (76.0%)	84 (84.0%)
200	53 (26.5%)	163 (81.5%)	84 (42.0%)	161 (80.5%)	173 (86.5%)
300	94 (31.3%)	254 (84.7%)	145 (48.3%)	251 (83.7%)	259 (86.3%)
400	134 (33.5%)	340 (85.0%)	217 (54.2%)	334 (83.5%)	350 (87.5%)
500	143 (28.6%)	427 (85.4%)	310 (62.0%)	420 (84.0%)	431 (86.2%)
600	176 (29.3%)	508 (84.7%)	381 (63.5%)	499 (83.2%)	516 (86.0%)
700	195 (27.9%)	582 (83.1%)	457 (65.3%)	576 (82.3%)	604 (86.3%)
800	224 (28.0%)	658 (82.3%)	516 (64.5%)	660 (82.5%)	681 (85.1%)
900	238 (26.4%)	725 (80.6%)	571 (63.4%)	724 (80.4%)	763 (84.8%)
1000	278 (27.8%)	795 (79.5%)	619 (61.9%)	795 (79.5%)	829 (82.9%)

Table B.1. Precision Results

location					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	54 (54.0%)	89 (89.0%)	67 (67.0%)	90 (90.0%)	84 (84.0%)
200	99 (49.5%)	175 (87.5%)	131 (65.5%)	171 (85.5%)	175 (87.5%)
300	142 (47.3%)	244 (81.3%)	191 (63.7%)	244 (81.3%)	257 (85.7%)
400	200 (50.0%)	303 (75.7%)	242 (60.5%)	300 (75.0%)	324 (81.0%)
500	237 (47.4%)	342 (68.4%)	304 (60.8%)	349 (69.8%)	371 (74.2%)
600	256 (42.7%)	387 (64.5%)	366 (61.0%)	385 (64.2%)	429 (71.5%)
700	293 (41.9%)	414 (59.1%)	422 (60.3%)	447 (63.9%)	474 (67.7%)
800	302 (37.8%)	430 (53.7%)	459 (57.4%)	485 (60.6%)	509 (63.6%)
900	306 (34.0%)	457 (50.8%)	498 (55.3%)		540 (60.0%)
1000	310 (31.0%)	475 (47.5%)	532 (53.2%)		
time					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	9 (9.0%)	28 (28.0%)	9 (9.0%)	28 (28.0%)	30 (30.0%)
200	13 (6.5%)	30 (15.0%)	9 (4.5%)	33 (16.5%)	33 (16.5%)
300	15 (5.0%)	30 (10.0%)	10 (3.3%)	33 (11.0%)	37 (12.3%)
400	18 (4.5%)	30 (7.5%)	11 (2.8%)	33 (8.3%)	37 (9.2%)
500	21 (4.2%)	32 (6.4%)	15 (3.0%)	34 (6.8%)	37 (7.4%)
600	23 (3.8%)	33 (5.5%)	16 (2.7%)	34 (5.7%)	40 (6.7%)
700	25 (3.6%)	33 (4.7%)	18 (2.6%)	34 (4.9%)	41 (5.9%)
800	25 (3.1%)	33 (4.1%)	18 (2.2%)	35 (4.4%)	43 (5.4%)
900	26 (2.9%)	37 (4.1%)	19 (2.1%)	35 (3.9%)	45 (5.0%)
1000	26 (2.6%)	37 (3.7%)	19 (1.9%)	35 (3.5%)	45 (4.5%)
weapon					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	23 (23.0%)	47 (47.0%)	32 (32.0%)	49 (49.0%)	42 (42.0%)
200	24 (12.0%)	59 (29.5%)	49 (24.5%)	63 (31.5%)	62 (31.0%)
300	28 (9.3%)	64 (21.3%)	60 (20.0%)	71 (23.7%)	73 (24.3%)
400	29 (7.2%)	64 (16.0%)	69 (17.2%)	77 (19.3%)	78 (19.5%)
500	29 (5.8%)	69 (13.8%)	70 (14.0%)	80 (16.0%)	85 (17.0%)
600	30 (5.0%)	69 (11.5%)	76 (12.7%)	82 (13.7%)	87 (14.5%)
700	31 (4.4%)	71 (10.1%)	79 (11.3%)	85 (12.1%)	87 (12.4%)
800	33 (4.1%)	71 (8.9%)	81 (10.1%)	85 (10.6%)	88 (11.0%)
900	33 (3.7%)	73 (8.1%)	84 (9.3%)		
1000	33 (3.3%)	73 (7.3%)	92 (9.2%)		

Table B.2. Precision Results

APPENDIX

RECALL RESULTS

Table C.1 presents recall data for all algorithms described in this thesis. The "Total" column represents the number of words generated. Recall is only shown for depths 100, 500, and "All". The "All" results represent recall at the end of 200 iterations of the algorithm, which may have generated as many as 1000 words. An algorithm may have generated fewer than 1000 words for reasons described in Section 4.1.2. The total number of category words in the corpus is listed next to each category's name in the table.

building (188 total words)					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	21 (11.2%)	35 (18.6%)	26 (13.8%)	34 (18.1%)	39 (20.7%)
500	33 (17.6%)	84 (44.7%)	86 (45.7%)	97 (51.6%)	100 (53.2%)
All	43 (22.9%)	92 (48.9%)	99 (52.7%)	102 (54.3%)	109 (58.0%)
event (501 total words)					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	61 (12.2%)	61 (12.2%)	64 (12.8%)	64 (12.8%)	61 (12.2%)
500	146 (29.1%)	196 (39.1%)	179 (35.7%)	200 (39.9%)	186 (37.1%)
All	190 (37.9%)	272 (54.3%)	245 (48.9%)	276 (55.1%)	266 (53.1%)
human (1856 total words)					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	36 (1.9%)	76 (4.1%)	39 (2.1%)	76 (4.1%)	84 (4.5%)
500	143 (7.7%)	427 (23.0%)	310 (16.7%)	420 (22.6%)	431 (23.2%)
All	278 (15.0%)	795 (42.8%)	619 (33.4%)	795 (42.8%)	829 (44.7%)
location (1018 total words)					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	54 (5.3%)	89 (8.7%)	67 (6.6%)	90 (8.8%)	84 (8.3%)
500	237 (23.3%)	342 (33.6%)	304 (29.9%)	349 (34.3%)	371 (36.4%)
All	310 (30.5%)	475 (46.7%)	532 (52.3%)	512 (50.3%)	564 (55.4%)
time (112 total words)					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	9 (8.0%)	28 (25.0%)	9 (8.0%)	28 (25.0%)	30 (26.8%)
500	21 (18.8%)	32 (28.6%)	15 (13.4%)	34 (30.4%)	37 (33.0%)
All	26 (23.2%)	37 (33.0%)	19 (17.0%)	35 (31.2%)	45 (40.2%)
weapon (147 total words)					
Total	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
100	23 (15.6%)	47 (32.0%)	32 (21.8%)	49 (33.3%)	42 (28.6%)
500	29 (19.7%)	69 (46.9%)	70 (47.6%)	80 (54.4%)	85 (57.8%)
All	33 (22.4%)	73 (49.7%)	92 (62.6%)	88 (59.9%)	88 (59.9%)

Table C.1. Recall Results

APPENDIX

CONFUSION RESULTS

Table D.1 and Table D.2 present confusion data for all algorithms described in this thesis. "Confusion" between categories refers to mistakes made when the algorithm chooses a word for a category although it really belongs to a different category. For example, if the word "Utah" is added to the *human* lexicon, then there is confusion between the *human* and *location* categories.

The data shown represent confusion after 500 words. Each major heading represents the category for which a word was added to the lexicon. The value of the "True Category" column represents the true category membership of the misclassified word. To use the example above, if BASILISK MCAT+ had chosen "Utah" for its *human* lexicon, then the value in the *human* table (BASILISK MCAT+ column, *location* row) would be incremented.

building					
True Category	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
event	36	9	14	12	9
human	25	48	77	33	38
location	235	207	93	108	89
time	32	1	10	7	5
weapon	1	5	1	0	1
event					
True Category	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
building	6	0	1	0	0
human	55	20	24	14	10
location	36	9	6	10	6
time	22	4	6	3	2
weapon	7	5	0	0	2
human					
True Category	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
building	10	2	1	2	1
event	28	2	8	3	3
location	26	3	9	5	6
time	18	3	3	3	2
weapon	5	3	1	4	2

Table D.1. Confusion Results

location					
True Category	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
building	16	14	9	6	3
event	53	24	11	9	6
human	20	23	33	22	22
time	34	5	16	5	3
weapon	1	1	0	0	0
time					
True Category	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
building	9	1	3	4	1
event	59	163	45	37	37
human	155	28	21	88	17
location	36	64	11	39	31
weapon	9	5	1	3	3
weapon					
True Category	METABOOT SCAT	BASILISK SCAT	METABOOT MCAT	BASILISK MCAT	BASILISK MCAT+
building	10	72	6	6	6
event	60	15	17	15	14
human	136	55	80	32	37
location	32	36	20	9	13
time	26	0	0	0	0

Table D.2. Confusion Results

REFERENCES

[1] *Proceedings of the Fourth Message Understanding Conference (MUC-4)* (San Mateo, CA, 1992), Morgan Kaufmann.

[2] AONE, C., AND BENNETT, S. W. Applying machine learning to anaphora resolution. In *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Understanding*. Springer-Verlag, Berlin, 1996, pp. 302–314.

[3] BLUM, A., AND MITCHELL, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning* (1998).

[4] CHARNIAK, E., ALTUN, Y., DE SALVO BRAZ, R., GARRETT, B., KOSMALA, M., MOSCOVICH, T., PANG, L., PYO, C., SUN, Y., WY, W., YANG, Z., ZELLER, S., AND ZORN, L. Reading comprehension programs in a statistical-language-processing class. In *Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*. Association for Computational Linguistics, 2000, pp. 1–5.

[5] DEMPSTER, A., LAIRD, N., AND RUBIN, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B 39* (1977), 1–38.

[6] DUNNING, T. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics 19*, 1 (1993), 61–74.

[7] GALE, W., CHURCH, K., AND YAROWSKY, D. A method for disambiguating word senses in a large corpus. *Computers and the Humanities 26* (1992), 415–439.

[8] GE, N., HALE, J., AND CHARNIAK, E. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora* (1998).

[9] HIRSCHMAN, L., LIGHT, M., BRECK, E., AND BURGER, J. D. Deep Read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics* (1999).

[10] KAMEYAMA, M. Recognizing referential links: An information extraction perspective. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics* (1997).

[11] KIM, J.-T., AND MOLDOVAN, D. I. Acquisition of semantic patterns for information extraction from corpora. In *Proceedings of the Ninth IEEE Conference on Artificial Intelligence for Applications* (Los Alamitos, CA, 1993), IEEE Computer Society Press, pp. 171–176.

[12] LENAT, D. B., PRAKASH, M., AND SHEPHERD, M. Cyc: Using common sense knowledge to overcome brittleness and knowledge-acquisition bottlenecks. *AI Magazine 6* (1986), 65–85.

[13] MARCUS, M., SANTORINI, B., AND MARCINKIEWICZ, M. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics 19*, 2 (1993), 313–330.

[14] MCCALLUM, A., AND NIGAM, K. Employing EM and pool-based active learning for text classification. In *Proceedings of the 15th Interational Conference on Machine Learning* (1998).

[15] MCCARTHY, J. F., AND LEHNERT, W. G. Using decision trees for coreference resolution. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (1995), pp. 1050–1055.

[16] MILLER, G. Wordnet: An on-line lexical database. In *International Journal of Lexicography* (1990).

[17] MOLDOVAN, D., HARABAGIU, S., PAŞCA, M., MIHALCEA, R., GOODRUM, R., GÎRJU, R., AND RUS, V. LASSO: A tool for surfing the answer net. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)* (1999).

[18] RILOFF, E. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (1996), AAAI Press/The MIT Press, pp. 1044–1049.

[19] RILOFF, E., AND JONES, R. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (1999).

[20] RILOFF, E., AND SCHMELZENBACH, M. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora* (1998).

[21] RILOFF, E., AND SHEPHERD, J. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing* (1997), pp. 117–124.

[22] RILOFF, E., AND THELEN, M. A rule-based question answering system for reading comprehension tests. In *Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*. Association for Computational Linguistics, 2000, pp. 13–19.

[23] ROARK, B., AND CHARNIAK, E. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics* (1998), pp. 1110–1116.

[24] SINGHAL, A., ABNEY, S., BACCHIANI, M., COLLINS, M., HINDLE, D., AND PEREIRA, F. AT&T at TREC-8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)* (1999).

[25] SODERLAND, S., FISHER, D., ASELTINE, J., AND LEHNERT, W. CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (1995), pp. 1314–1319.

[26] SRIHARI, R., AND LI, W. Information extraction supported question answering. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)* (1999).

[27] THOMPSON, C. A., AND MOONEY, R. J. Automatic construction of semantic lexicons for learning natural language interfaces. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (1999).

[28] VOORHEES, E. M., AND TICE, D. M. The TREC-8 question answering track evaluation. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)* (1999).

[29] YAROWSKY, D. One sense per collocation. In *Proceedings, ARPA Human Language Technology Workshop* (1993), Princeton, pp. 266–271.

[30] YAROWSKY, D. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* (Cambridge, MA, 1995), pp. 189–196.

[31] ZELLE, J. M., AND MOONEY, R. J. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (1996), AAAI Press/The MIT Press, pp. 1050–1055.