Automatic Lexicon Generation
for Unsupervised Part-of-Speech Tagging Using Only Unannotated Text

Dennis V. Pereira

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Computer Science

Dr. Csaba Egyhazy, Chair
Dr. William Frakes
Dr. Gabriella Belli

August 13, 2004
Falls Church, VA

Keywords: automatic, lexicon, lexicon generation, part-of-speech, term categorization
Automatic Lexicon Generation
for Unsupervised Part-of-Speech Tagging Using Only Unannotated Text

Dennis V. Pereira

Abstract

With the growing number of textual resources available, the ability to understand them becomes critical. An essential first step in understanding these sources is the ability to identify the parts-of-speech in each sentence. The goal of this research is to propose, improve, and implement an algorithm capable of finding terms (words in a corpus) that are used in similar ways – a term categorizer. Such a term categorizer can be used to find a particular part-of-speech, i.e. nouns in a corpus, and generate a lexicon. The proposed work is not dependent on any external sources of information, such as dictionaries, and it shows a significant improvement (~30%) over an existing method of categorization. More importantly, the proposed algorithm can be applied as a component of an unsupervised part-of-speech tagger, making it truly unsupervised, requiring only unannotated text. The algorithm is discussed in detail, along with its background, and its performance. Experimentation shows that the proposed algorithm performs within 3% of the baseline, the Penn-TreeBank Lexicon.
Automatic Lexicon Generation
for Unsupervised Part-of-Speech Tagging Using Only Unannotated Text
(August 2004)

Dennis V. Pereira

TABLE OF CONTENTS

I. Introduction ...................................................................................................................1
   A. Good vs. "cheap" words ..............................................................................................1
   B. Natural Language Processing ....................................................................................1
   C. Hypothesis ..................................................................................................................2

II. Literature Review..............................................................................................................2
   A. Define Parts-of-Speech ...............................................................................................2
      1) Types of Words ..........................................................................................................3
         a) Words with Unambiguous Parts-of-Speech ..........................................................3
         b) Unambiguous Usage ..............................................................................................3
         c) Ambiguous Usage – Need for Outside Knowledge ................................................3
      2) Foundation Part-of-Speech Computation .................................................................3
      3) Using Local Context – "yarzbygu" Example .............................................................3
      4) Part-of-Speech Computation ..................................................................................4
         a) Multi-Lingual ...........................................................................................................4
         b) Written English Focus ..........................................................................................4
         c) Beyond Written Language ....................................................................................4
   B. Lexicon .......................................................................................................................5
      1) Need For Lexicon .....................................................................................................5
         a) POS Tagger Architecture ......................................................................................6
      2) Sample Lexicon – giraffe example ..........................................................................6
      3) Cost of Creating a Lexicon .....................................................................................6
      4) Penn-TreeBank ........................................................................................................7
      5) Limitations of Humans Creating Lexicons ............................................................7
   C. POS Taggers ...............................................................................................................7
      1) Brill Taggers – Transformation Based Approach ....................................................7
         a) Supervised Tagger ..................................................................................................8
         b) Unsupervised Tagger ...........................................................................................8
         c) Brill Tagger Pitfall ................................................................................................8
      2) Statistical POS Taggers ..........................................................................................9
         a) Zipf’s Law .............................................................................................................9
         b) MXPOST – N-Gram Approach ..........................................................................9
         c) TNT – HMM Approach ......................................................................................9
         d) BNC – Template Approach ...............................................................................10
      3) Autotutor – Neural Network Approach ................................................................11
      4) Approach Commonality .......................................................................................11
      5) Intro to Unknown Word Guessing .........................................................................11
   D. Unknown Word Guessing .......................................................................................11
      1) Context and Morphology .......................................................................................12
      2) Unknown Word Guessing Methods .....................................................................12
         a) Mikheev’s Approach .........................................................................................12
            (1) Framing the Problem .....................................................................................13
            (2) Naïve and Simple Approaches .....................................................................13
            (3) Comparing Mikheev and Brill .....................................................................13
         b) Thede’s Statistical Approach .............................................................................14

(3) Comparing Mikheev and Brill .............................................................................13
   2) Statistical POS Taggers ..........................................................................................9
      a) Zipf’s Law .............................................................................................................9
      b) MXPOST – N-Gram Approach ..........................................................................9
      c) TNT – HMM Approach ......................................................................................9
      d) BNC – Template Approach ...............................................................................10
   3) Autotutor – Neural Network Approach ................................................................11
   4) Approach Commonality .......................................................................................11
   5) Intro to Unknown Word Guessing .........................................................................11
   D. Unknown Word Guessing .......................................................................................11
      1) Context and Morphology .......................................................................................12
      2) Unknown Word Guessing Methods .....................................................................12
         a) Mikheev’s Approach .........................................................................................12
            (1) Framing the Problem .....................................................................................13
            (2) Naïve and Simple Approaches .....................................................................13
            (3) Comparing Mikheev and Brill .....................................................................13
         b) Thede’s Statistical Approach .............................................................................14
V. Conclusion........................................................................................................................................34
    A. Future Work.................................................................................................................................34
References ..........................................................................................................................................36
Appendix A .........................................................................................................................................37

Table of Figures
Figure 1: Research Architecture......................................................................................................2
Figure 2: Approaches for POS Tagging [34]....................................................................................4
Figure 3: Traditional POS-Tagger Architecture [25].......................................................................6
Figure 4: Brill’s description of transformation-based rule learning [8].............................................8
Figure 5: Cucerzan’s example of a prefix trie for “Alex and Anda are a nice couple” [13]..........10
Figure 6: Term Weighting Equation [36]........................................................................................15
Figure 7: Defining Accuracy [21].....................................................................................................16
Figure 8: Comparing POS taggers using word-only to predict parts-of-speech [21]......................18
Figure 9: Research Process – Control Case, Benchmark, and Experimentation ...............................22
Figure 10: Lexicon Generation Algorithm.......................................................................................27

Table of Tables
Table 1: Open-class tags [22]...........................................................................................................13
Table 2: POS Tagger Accuracy [21].................................................................................................17
Table 3: POS Ambiguity Schemes – Showing adjectives and adverbs have high error rates [25]....18
Table 4: Top 20 N-Graphs ..............................................................................................................25
Table 5: Discriminant Analysis Classification Results......................................................................30
Table 6: Baseline Algorithms vs. Lexicon Generator......................................................................32
Table 7: Brill Tagger Accuracy Given Different Configurations of the Lexicon..............................32
Table 8: 10-Fold Validation and Lexicon Size .................................................................................33
**Abstract**— With the growing number of textual resources available, the ability to understand them becomes critical. An essential first step in understanding these sources is the ability to identify the parts-of-speech in each sentence. The goal of this research is to propose, improve, and implement an algorithm capable of finding terms (words in a corpus) that are used in similar ways—a term categorizer. Such a term categorizer can be used to find a particular part-of-speech, i.e., nouns in a corpus, and generate a lexicon. The proposed work is not dependent on any external sources of information, such as dictionaries, and it shows a significant improvement (~30%) over an existing method of categorization. More importantly, the proposed algorithm can be applied as a component of an unsupervised part-of-speech tagger, making it truly unsupervised, requiring only unannotated text. The algorithm is discussed in detail, along with its background, and its performance. Experimentation shows that the proposed algorithm performs within 3% of the baseline, the PennTreeBank Lexicon.

**Index Terms**— automatic, lexicon, lexicon generation, part-of-speech, term categorization

I. INTRODUCTION

A. Good vs. “cheap” words

“Why is one word good and another word cheap? … The question was confronted by the editors of a brand-new dictionary, The American Heritage Dictionary, at the outset of their task in the mid-1960s. They assembled a “Usage Panel” to help them appraise the new words and dubious constructions that had come knocking at the door. Which ones should be ushered in, which thrown out on their ear? The panel consisted of 104 men and women—mostly writers, poets, editors and teachers—who were known for caring about the language and trying to use it well” [38].

In William Zinsser’s book [38] “On Writing Well” an entire chapter is devoted to correct usage of the English language. Seventy-five percent of the chapter focuses on the work necessary to decide if a word and its usage should be included in The American Heritage Dictionary. Zinsser gives an example: “Would I allow “like” to be used as a conjunction—like so many people do? How about “mighty,” as in “mighty fine”?” [38]?

We can clearly see that agreeing on the use of terms in American English is not a trivial task. However, people will undoubtedly use terms as they wish. Zinsser puts emphasis on this problem by saying “any dolt can rule that the suffix “wise,” as in “healthwise,” is doltwise, or that being “rather unique” is no more possible than being rather pregnant” [38]. He speaks of himself as the dolt, or stupid person, who decides how terms will be used and described in the dictionary, but he continues by saying that it is his duty (along with the other 103 panel members) to allow the language to grow in strength and in color [38].

In any given population new terms are guaranteed to be introduced. These could be for good reasons such as a need that did not previously exist, or they could be for bad reasons such as covering up scandal by embellishing the terminology used to describe it. Meritorious, indeed, is the effort by Zinsser and his colleagues to establish a convention by which all American English speaking people can base their communication. But, in a world of rapid change, and constant communication, people are becoming lazier. And because of the quantity, ease, and need for daily communication, their methods of expression become distorted and they create their own words with their own meanings. Such laziness leads to the use and adoption of words that are not in any dictionary. Words that Zinsser regards as the ones that may strengthen and color the language, but may also weaken the language.

Firth quoted by Zhai [37] said, “You shall know a word by the company it keeps.” How true this statement holds for both Zinsser and for the child or student learning a language. Clark [12] discusses how infant children acquire language, saying “at this early phase of learning, only limited sources of information can be used: primarily distributional evidence, about the contexts in which words occur, and morphological evidence...about the sequence of symbols (letters or phonemes) of which each word is formed.” Clark re-affirms Firth’s quote that a word is known by its context and then adds an additional statement that a word is also known by its structure. But, how can a computer take advantage of these characteristics in order to learn language just as a child?

B. Natural Language Processing

Machines face the same problems as humans when attempting to understand an unknown language. In fact, there is a field of computer science dedicated to the understanding of human language called natural language processing, or NLP.

In Natural Language Processing, identifying the parts-of-speech is a critical step towards being able to produce a variety of useful products. These can include a thesaurus for information retrieval purposes [30]; the ability to perform information extraction [13]; the ability to perform machine translation [24]; the ability to retrieve text across languages [36]; and a host of others.

Megyesi [21] stresses the importance of correct part-of-speech annotation in text to speech systems by showing that a different part-of-speech for the same word can drastically change its vocalization. Vasilikopoulos [35] mentions that errors in part-of-speech tagging lead to larger, more significant errors downstream, “when processing huge amounts of data even a very small error rate of 3-4% introduces approximately one error per sentence and so the propagation of these errors grows more than linearly henceforth. Moreover, because of this fact, other NLP tasks such as word sense disambiguation, question answering, information retrieval, etc. which rely on the trivial ones cannot perform very well.”

The process of identifying the parts-of-speech with a machine is done by systems called part-of-speech taggers, or
POS taggers. This process of assigning parts-of-speech involves three steps: tokenization – the separation of terms; term categorization – determining the possible uses of each term; and term disambiguation – resolving a single usage category for a term from its categorical possibilities [17]. While the focus of most parts-of-speech taggers has been on step three – term disambiguation, the focus of this paper is on step two – term categorization. The importance of term categorization will become apparent after reading the survey of taggers and their deficiencies. Taggers come in a variety of flavors, from ones that statistically determine parts-of-speech to others that learn rules to identify the parts-of-speech. Although their approaches may be vastly different, the most prominent approaches for POS tagging are dependent on a dictionary-derived lexicon (as defined in Section II. B.) to provide the necessary term categorization.

C. Hypothesis

When done manually, part-of-speech tagging is a labor-intensive task requiring substantial time and financial resources. Unfortunately, there is no single, automatic way to retrieve the parts-of-speech from an arbitrary document in any language, this is because existing POS taggers depend on language-specific dictionaries of terms (lexicons). The hypothesis presented in this paper is that by using existing methods, it is possible to generate a lexicon directly from the text needing to be processed, without using any external information or lists. The proposed algorithm automatically assigns terms to a class of tags based on grammatical markers in the text, such as context, word frequencies, and word morphology. The dependent and independent variables are identified in a subsequent section (Section III. C. – Components). The goal of this research is to define and implement a framework for automatically categorizing terms into similar parts-of-speech, in other words, automatic generation of the lexicon. Term categorization is not restricted to parts-of-speech, it can also be used for named entity recognition and unknown word guessing, but extensive work done with parts-of-speech provides a robust set of existing data, including hand tagged text that can be used for verification purposes. With these advantages, using the parts-of-speech provides a strong basis of comparison against other methods.

In this paper automatic lexicon generation is addressed for a subset of American English. The ultimate goal is to be able to automatically generate a lexicon for any language given a set of documents (a corpus) in that language. Unfortunately, because the existing data used for comparison is American English, reliable comparisons can only be done in this limited scope. The secondary hypothesis is that by automatically generating a lexicon, errors are introduced into the learning (or training) process of the part-of-speech tagger; but the tagger can overcome the errors introduced by the automatically generated lexicon and continue to produce results that are comparable to using a dictionary-derived lexicon. The dependent and independent variables are identified in a subsequent section (Section III. C. – Components).

The approach taken to test the hypotheses is multi-tiered. The first task is to create a lexicon generator whose results are then used as input to a POS tagger. The output of the POS tagger is then compared with the output of the POS tagger given a different, pre-existing, lexicon. Looking at figure 1 the following question could be asked: a system C has inputs A and B, how does the output of system C compare given these different inputs? System C is the part-of-speech tagger. Input A is a pre-existing control lexicon, considered to be the correct answer. Input B is the generated lexicon, created by the framework and algorithm described in this paper. The various parts of Figure 1 are discussed throughout this paper, highlighting the reasoning behind existing methods, their deficiencies, and the proposed method for addressing the problem. The results, presented later, show the usefulness of a generated lexicon in the context of part-of-speech tagging.

Figure 1: Research Architecture

II. LITERATURE REVIEW

A. Define Parts-of-Speech

We begin by identifying how parts-of-speech are recognized, how they are categorized, and how they are annotated. Identifying parts-of-speech may seem like a trivial task at first, but it quickly becomes a difficult problem, and is discussed shortly. Humans have the ability to identify and distinguish among written words in many ways. Three such ways are:
1. first, by reading a statement and understanding the meaning because the words selected are unambiguous,
2. second, by reading a statement and understanding the meaning because the words selected are used in an unambiguous manner, and
3. third, by using knowledge outside the written statement to deduce the meaning.

1) Types of Words

a) Words with Unambiguous Parts-of-Speech

When the individual words used in a statement are unambiguous, identifying the parts-of-speech is straightforward, take the following statement for example:

This car is big

Each word in this statement has exactly one part-of-speech, and although the sentence as a whole may be ambiguous, the part-of-speech assignment for each word in the statement is unambiguous. No matter how the words are rearranged they will always have the same part-of-speech.

This/DT car/NN is/VB big/JJ

DT = Determiner
NN = Noun
VB = Verb
JJ = Adjective

b) Unambiguous Usage

If every statement contained only words that had exactly one part-of-speech, there would be no reason to continue. However, there are several ways this statement could be modified to make it more difficult to determine each word’s part-of-speech. For example, if the word “big” is replaced with the word “slow” then there would no longer be exactly one way to assign parts-of-speech to the sentence: The statement would read:

This car is slow

Since we are now using “slow” instead of “big” we must consider the other ways “slow” can be used, for example, “Slow this car down.” The word “slow” can be used as a verb, which means there are now two options for assigning the parts-of-speech to the sentence:

1. This/DT car/NN is/VB slow/JJ
2. This/DT car/NN is/VB slow/VB

The difference between the two options is “slow” as an adjective (JJ) and “slow” as a verb (VB). Fortunately, the word “slow” in this statement is unambiguous, because it is clear that “slow” describes the car and not an action the car is performing; therefore, “slow” cannot be a verb in this statement. This an example of selecting words that are used in an unambiguous manner.

Another example of a word that has more than one part-of-speech, but is used unambiguously is given below. The following two statements contain the word “show,” however each statement uses the word in a different part-of-speech. The first sentence uses “show” as a verb, while the second statement uses “show” as a noun.

1. Surveys show that one out of three Americans
2. During the five-day show

Below are the correctly annotated versions of these statements:

1. Surveys/NN show/VB that/IN one/CD out/IN of/IN three/CD Americans/NN
2. During/IN the/DT five-day/JJ show/NN

IN = Preposition
CD = Cardinal Number

Here we see that the part-of-speech for the word “show” is unambiguous in both cases because of the order and selection of words used in the statement. However, the words selected for some statements leave the meaning to be determined by the reader.

c) Ambiguous Usage – Need for Outside Knowledge

The hardest case for determining parts-of-speech for a statement is when the same terms, in the same order, can have multiple meanings. In these types of situations it is often difficult, even for a human, to determine the correct usage from the written statement and requires outside knowledge to determine the correct interpretation.

An example,

The sentence “time flies like an arrow” can be interpreted in at least the following ways:

1. Time passes along in the same manner as an arrow gliding through air;
2. I order you to take timing measurements on flies, in the same manner as you would time an arrow;
3. Fruit flies like to feast on a banana; in contrast, the species of flies known as “time flies” like an arrow;

Clearly only the first of these interpretations would be considered by a competent reader although they are all valid syntactical interpretations of the sentence. We choose the first interpretation using extra information about time, flying and arrows to choose the most sensible interpretation in the given context [15].

2) Foundation Part-of-Speech Computation

In the previous three sections it has been demonstrated that (1) a statement may contain only terms whose part-of-speech is unambiguous, (2) a statement may contain terms that have multiple parts-of-speech, but whose part-of-speech is unambiguous due to its usage, and (3) that a statement can contain terms that make it completely ambiguous requiring outside knowledge to interpret. Cases one and two demonstrate the ability of a human to understand a statement from the text itself, while case number three is unique and will not be considered. It is on the assumption that the parts-of-speech can be derived from the text that the idea of computationally deriving the parts-of-speech is founded.

3) Using Local Context – “yarzygu” Example

One of the ways a human is able to understand a statement is by using local context. Each word has other words
surrounding it. If the words of a statement are ordered correctly, then the statement will be understood. Even if one of the words is unknown, the context gives strong evidence as to the part-of-speech role the word plays in the statement. To provide an example, an unknown word is selected by creating a pronounceable sequence of letters – “yarzbygu.” If this word is inserted into the following sentence, “tomorrow we will go to the yarzbygu to buy some toys,” a human can tell that “yarzbygu” is a noun, probably a toy store. No matter what unknown word is selected for this example, the part-of-speech will always be the same because of its local context. If “yarzbygu” is inserted into a different sentence – “I yarzbygu those toys.” Then the context makes this word a verb. Even with a word that plays multiple parts-of-speech roles, if a machine is given a sufficient number examples containing the different uses of the word, it can begin to learn the proper part-of-speech for that word by using its context.

4) Part-of-Speech Computation

The computational term for assigning parts-of-speech to a word is called Part-of-Speech (POS) tagging. POS tagging has been a heavily researched topic, approached from many different angles and resulting in many different styles of taggers, all of which produce satisfactory results. In the best conditions these taggers are capable of 97% accuracy on standard English corpora, such as the Brown corpus (see section III. C. 2. and IV. C.). Machine learning has allowed the field of Natural Language Processing (NLP) to use relatively small amounts of data to analyze and annotate large amounts of text. This allows a significant amount of data to be processed in a very short amount of time. Megyesi [21] put it this way, “one of the most popular NLP areas that machine learning algorithms have been successfully applied to is POS tagging, i.e. the annotation of words with the contextually appropriate POS tags…The average accuracy that are reported for state of the art data-driven POS taggers lies between 95% and 98% depending on the language type the taggers are trained and tested on” [21].

It is important to identify the various means of creating a part-of-speech tagger. The graph in Figure 2 depicts the different approaches to tagging parts-of-speech. There are two primary ways of attacking the problem – in a supervised or an unsupervised fashion. A supervised approach involves a human who helps the tagger achieve the final answer (Figure 2 shows the Hidden Markov model as the common link across all supervised stochastic approaches). On the other hand, an unsupervised approach involves specialized algorithms that can learn from examples and achieve an answer without the help of a human. Both supervised and unsupervised methods have reported accuracy in the upper 90th percentile.

Figure 2: Approaches for POS Tagging [34]

a) Multi-Lingual

“The main advantage with data-driven POS taggers is that they are language and tag set independent and thereby are easily applicable to new languages and domains” [21]. However, “the applicability of AI-style algorithms and supervised methods is limited in the multilingual case because of the cost of knowledge databases and manually annotated corpora” [13]. The previous two statements may seem contradictory, but they are not, they are both incomplete expressions of the same topic. POS taggers have the advantage that they can be run on multiple languages; unfortunately, no matter which type of tagger is selected (supervised or unsupervised), language dependent data are required. This language dependent data may be statistics for the desired language, a sample document of the desired language containing appropriately tagged words to be used as training, and/or a dictionary of the desired language. The taggers that claim to be language independent do indeed function on any language, given these added pieces of data. However, the cost of creating this additional data, in many cases, cannot be justified.

b) Written English Focus

Additionally, it is worth mentioning that the majority of studies have focused on written English because supplementary data such as dictionaries and previously tagged documents are readily available. Megyesi [21] puts this claim in the scope of data chunking, saying “the majority of studies on chunking has been focused on the development of data-driven chunkers/parsers for English, just as it was in the case of part-of-speech tagging task a couple of years ago. The reason is mainly that there is a correctly parsed corpus for English, the Penn TreeBank, while such a corpus is missing for most of the languages. Given this ‘correctly’ parsed large data set, the development and evaluation of the data-driven approaches become easier and reliable” [21].

c) Beyond Written Language

An interesting problem arises once the bounds of the formal studies on written languages are crossed. A team from
the University of Memphis has developed a system called AutoTutor [24], an interactive educational tutoring machine. The AutoTutor team has identified that "the language of many learners...is more akin to oral conversation than to printed text. Much of the language is ungrammatical, vague, semantically ill-formed, incoherent, and replete with repairs and metacommunication markers (e.g., uh-huh, uh)" [24]. Thus, when the problem of part-of-speech tagging extends beyond the written language and into the spoken language, the capability of existing resources is significantly reduced and the challenge in processing this type of information becomes much more difficult.

The typical approach to POS tagging requires: a set of documents in need of tagging, a dictionary of the corresponding language, and an algorithm that makes use of linguistic phenomena such as word frequency, context, morphology, and syntax. The details of these various approaches are discussed in a later section. The most critical part of the POS tagging process is the set of documents – a corpus. A corpus is necessary to provide the POS tagger the types of sentences expected to be seen in the future. The corpus used in POS tagging is typically broken into two sets; first, a training set, which is used to instruct the system on what types of language it is expected to process, and second, a test set, which is used to run and evaluate the system. The training set is significantly larger than the test set, as Brants [7] mentioned in his experiments, “all tests are performed on partitions of the corpora that use 90% as training set and 10% as test set, so that the test data is guaranteed to be unseen during training.” Similarly, Branco and Silva [6] evaluated their “approach by training the tagger over 90% of the corpus...[using an] evaluation corpus with the remainder 10%, obtained by extracting one out of each 10 consecutive sentences.”

B. Lexicon

Together with a corpus, a computer can begin to understand a natural language by using a lexicon. A lexicon can be described as a dictionary, however, in this paper the definition is refined to be a set of words derived from a dictionary along with their corresponding parts-of-speech. A lexicon may contain multiple parts-of-speech for each word, but it may only contain a single instance of each word. Because the focus of this work is on English text, a word is defined as a case-sensitive set of characters that does not contain any white space. This means that the same word capitalized differently has a separate entry in the lexicon. The most common instance of these types of words can be found at the beginning of a sentence, such as "school" and "School" in the example sentences:

I love school.
School is great!

However, allowing entries for words with differing capitalization does not mean that a word with the same capitalization has multiple entries when it has more than one part-of-speech. Take for example, the word “writing”, it appears only once in the lexicon, but “writing” can be used as both a noun and a verb. The lexicon contains a single entry for “writing” containing its two parts-of-speech. To better understand the appearance of a lexicon entry, the entry for the word “writing” is shown:

writing VBG NN

This entry shows that “writing” can be used as either a verb (VBG) or a noun (NN). An explanation of the POS tags is provided in later sections.

Some lexicons contain additional information, including counts or probabilities of occurrence. Thede [33] has provided an example of the type of lexicon used in his experiments, “the lexicon entry for the word advanced is the following:

advanced ((VBN 31) (JJ 12) (VBD 8))

This means that the word advanced appeared a total of 51 times in the corpus: 31 past participles (VBN), 12 adjectives (JJ), and 8 past tense verbs (VBD)” [33]. Since each instance of “advanced” was represented in the lexicon, the probability of occurrence can be calculated. Thede [33] explains that “this lexicon gives P(t|w), which can then be used to calculate P(w|t)= P(w) P(t|w) / P(t).” He emphasizes that “part-of-speech tagging depends on a lexicon of words to supply the required P(w|t). If a word is not available in the lexicon, then this probability needs to be provided in some other way” [33]. The probability P(t|w) mentioned above indicates that the lexicon provides the probability that a tag appears given a certain word. Thede’s [33] equation to calculate P(w|t|, can be defined by multiplying the probability of a word appearing with the probability of a tag given that word, and then dividing the product by the probability of the tag. P(w) is a simple calculation performed by counting the instances of the word and dividing by the total number of words. P(t|w) is provided by the lexicon, for the word “advanced” in this example, P(t|w) for VBN = 31/51 = .61, for JJ = 12/51 = .24, for VBD = .16. And, P(t) can be calculated as simply as P(w), by counting the instances of the tag and dividing by the total number of words.

1) Need For Lexicon

Given Thede’s equation and need for calculating P(w|t|, it becomes more evident how a lexicon helps a computer process natural languages. However, let us establish a firm need for the lexicon before understanding how it can be used. Zhai [37] identifies the need for lexicons and emphasizes their importance in modern grammar theories by stating that “many modern grammar theories are now converging on the acceptance of the increasingly important role of lexicon.” Orphanos [25] makes the claim that “although the hardest part of the tagging process is performed by a computational lexicon, a POS tagger cannot solely consist of a lexicon.” And, Olde [24] takes the next step when describing his system; the AutoTutor system “first consults a lexicon to identify the set of possible tags for each word, then uses a neural network to select a single tag for each word.”
Zhai [37] has properly identified the need for lexicons in modern grammar theories and many systems rely on lexicons to provide the basic information necessary to produce larger and more useful results. The following “tagger architecture”, provided by Orphanos et. al [25], in Figure 3, is common to almost all part-of-speech taggers.

**Figure 3: Traditional POS-Tagger Architecture [25]**

Orphanos [25] describes the figure beginning with, “raw text passes through the Tokenizer, where it is converted to a stream of tokens. Non-word tokens (e.g., punctuation marks, numbers, dates, etc.) are resolved by the Tokenizer and receive a tag corresponding to their category. Word tokens are looked-up in the Lexicon and those found receive one or more tags. Words with more than one tag and those not found in the Lexicon pass through the Disambiguator/Guesser, where the contextually appropriate tag is decided/guessed.”

2) **Sample Lexicon – giraffe example**

Now that we understand the structure, role, and importance of the lexicon, let us take a look at a larger sample of a lexicon. Below is an example of such a lexicon:

- Quizzical NNP
- wife-to-be NN
- ogress NN
- Charity NN NNP
- forearm NN
- British JJ NNP NNS NNPS
- MMS NNP
- Dickey NNP
- Democrats NNPS NNP NNS VBP
- Packet NN
- Seat NN
- temporary JJ
- Drugstore NNP NN
- Hindelong NNP
- Seiler NNP
- plain-spoken JJ
- bucked VBD VBN
- leviathan JJ NN
- introspective JJ
- Likewise RB
- Underground JJ
- Financieros NNP
- novelists NNS
- singlehandedly RB
- Bars NNP
- experimentation NN
- rating NN VBG
- flipping VBG JJ RB
- endurable JJ
- fly-by-nighters NNS
- evergreens NNS
- Deyo NNP
- fiscal JJ IN NN
- butchers NNS VBZ
- physiological JJ
- self-appointed JJ VBN
- demagogues NNS
- chisel NN VB
- anti-conservation JJ
- poems NNS

Given this list of words and their possible parts-of-speech we can see that some words can fall into more than one category. But, each word when used in a written document can only have a single part-of-speech for each occurrence, unless its meaning is truly ambiguous.

Thinking back to our days in elementary school, we too identified the parts of speech (noun, verb, adjective, adverb) in order to learn how to understand the meaning of a sentence. In our own minds, we have a type of lexicon that puts words into certain categories. For example, it would be uncommon to use the word “giraffe” as a verb. “Can someone be giraffed?” Or, can someone be told to “giraffe that thing?” These sentences do not make sense, and subconsciously, before we even speak or write them, we know they don’t make sense because of our ability to quickly and automatically put words into their correct categories of usage.

3) **Cost of Creating a Lexicon**

Unfortunately, when compared with the direct monetary cost of creating a lexicon in our mind, creating a lexicon for a computer is very expensive and time consuming. The advantage of creating a lexicon for a computer is its capability to retain much more information than a person; for example, a computer is capable of retaining a lexicon for every natural language. But, creating such lexicons are multi-year efforts that cost significant amounts of money.

Typically, a lexicon is produced by manually tagging a corpus. Once a corpus has been manually tagged a simple program creates the lexicon using the rule that a word shall
have a single entry containing a list of its part-of-speech role(s). One such effort for English is called the PennTreeBank [20].

4) Penn-TreeBank

Headed by the University of Pennsylvania, the Penn-TreeBank was an effort to develop a manually tagged corpus that could be used as a baseline for learning parts-of-speech in the English language. The project included a corpus consisting of 4.5 million words and a set of 36 parts-of-speech tags along with 12 additional tags for punctuation and currency [20]. The corpus was made up of articles from the WallStreet Journal combined with a previously built corpus called the Brown corpus.

In beginning the process of creating the Penn-TreeBank, Marcus [20] explained that one of the first decisions to be made was the level of granularity for the parts-of-speech tags. Since some of the documents proposed for use in this corpus came from previously tagged corpora, Marcus had to decide between keeping one of the existing tag sets or creating an entirely new set. Either choice would have required significant effort to convert the existing tag set into another. Marcus explained it this way:

The POS tagsets used to annotate large corpora in the past have traditionally been fairly extensive. The pioneering Brown Corpus distinguishes 87 simple tags ([Francis 1964]), ([Francis and Kucera 1982]) and allows the formation of personal pronoun; thus, the contraction ”I'm” is tagged as PPSS+BEH (PPSS for “non-3rd person nominative personal pronoun” and BEM for “am, 'm”). Subsequent projects have tended to elaborate the Brown Corpus tagset. For instance, the Lancaster-Oslo/Bergen (LOB) Corpus uses about 135 tags, the Lancaster UCREL group about 165 tags, and the London-Lund Corpus of Spoken English 197 tags. The rationale behind developing such large, richly articulated tagsets is to approach “the ideal of providing distinct codings for all classes of words having distinct grammatical behaviour” (Garside et al 1987, 167) [20].

Obviously, the approach taken by the groups listed by Marcus is to expand the sets to be as large and complete as possible. The Penn-TreeBank is very different, it has fewer than half of the number of tags identified by the Brown Corpus, and it has only 20-30% the number of tags of the other corpora.

5) Limitations of Humans Creating Lexicons

A significant point needs to be made that the Penn-TreeBank is a large effort to create a lexicon for a single language. Additionally, by annotating a corpus, the lexicon created from that corpus is limited to the words appearing in the selected documents. Marcus et. al [20] established a method of manually tagging a corpus and described the level-of-effort needed to annotate corpora for parts-of-speech in the following way: “The learning curve for the POS tagging task takes under a month (at 15 hours a week), and annotation speeds after a month exceed 3,000 words per hour” [20].

In order to explain the time and cost of this commitment, each person requires a month’s worth of training to learn how to correctly use the system and the rules for annotating words for the Penn-TreeBank. At that rate “…a team of five part-time annotators annotating three hours a day should maintain an output of about 2.5 million words a year of “tree banked” sentences…” [20]. Given that the Penn-TreeBank consists of 4.5 million words, at this rate it would take almost 2 years to annotate the Penn-TreeBank corpora. Surely there must be an easier and more efficient, perhaps even automated, way to perform this task.

Even with a group of annotators, a human’s ability to annotate faster will reach a finite limit. Once that limit has been reached, there is nothing else that human, or group of humans, can do to resolve the problem any faster. It is useful, however, to have a manually tagged corpus for the sake of creating a baseline by which all systems can be evaluated. But for a time critical project, where some margin of error can be tolerated, a year is too long to wait. In some instances the information is needed in only a few hours. Being able to provide an answer quickly can mean the difference between a multi-million dollar sale and no sale at all. If we are able to produce results similar to the system considered to be “correct,” then a great accomplishment has been made.

C. POS Taggers

Orphanos [25] said that “according to the data-driven approach, a frequency-based language model is acquired from corpora and has the forms of n-grams, rules, decision trees, or neural networks.” Addressing Orphanos’ claim, this section focuses on the role of a lexicon in three types of taggers: transformation based, stochastic/statistical, and neural network. Five different systems are presented including: Brill, MXPOST, TNT, BNC (CLAWS4), and AutoTutor.

1) Brill Taggers – Transformation Based Approach

Brill taggers come in two flavors, a supervised [9] and an unsupervised [10] version. Both are used for identifying parts of speech—including Adjectives, Modals, Singular Proper Nouns, Singular or Mass Nouns, Possessives, Verb Base Forms, Verbs Past Tense, Verbs Past Participle, Verbs Non-3rd Person Singular Present, and so on. A complete list is presented later.

Megyesi [21] describes the Brill tagger in a general sense, calling it “a rule-based approach that learns by detecting errors. It begins with an unannotated text that is labeled by an initial-state annotator in a heuristic fashion. Known words (according to some lexicon) are annotated with their most frequent tag while unknown words receive an initial tag (e.g. the most frequently occurring tag in the corpus). Then, an ordered list of rules learned during training are applied deterministically to change the tags of the words according to their contexts. Unknown words are first assumed to be nouns and handled by prefix and suffix analysis by looking at the first/last one to four letters, capitalization feature and adjacent word co-occurrence. For the disambiguation of known words, TBL [Transformation Based Learning] uses a context of up to
three preceding and following words and/or tags of the focus word as default” [21].

Megyesi [21] did a good job of describing both versions of the Brill tagger in a single paragraph, along with addressing the additional features used to categorize unknown words. Brill chose to use context and characteristics of the words – morphology. In the sections that follow, it is shown that context and morphology are the two most commonly used attributes for performing POS tagging.

a) Supervised Tagger

Brill [9] devised a supervised tagger in 1994, dependent on a manually tagged corpus to indicate the correct part-of-speech for each word. Brill uses this manually tagged corpus to create a lexicon. Using the information collected in the lexicon, the tagger applies the most likely tag to each word. It then applies transformations (example below) to the set of tags to reduce the error between the generated rule and the correct answer. Once the tagger can no longer reduce the error rate by a predetermined amount, it terminates. The result of this process is a set of rules that can be used to identify the parts-of-speech by performing transformations on other untagged sets of data. These rules can be applied to new sets of data without the need for a new manually tagged set.

This approach is called supervised because it is dependent on a manually tagged training corpus, requiring a human to read and annotate each word with its correct part-of-speech. This correctly tagged corpus is used to generate a lexicon. It is also used to compare against the tagger’s resulting annotated corpus to derive transformation-based rules. An example of such a transformation rule is:

Change the tag of a word from VERB to NOUN if the previous word is a DETERMINER.[10]

These transformations are applied recursively to get better rules until a threshold is met or no more rules can be produced. Figure 4 describes Brill’s algorithm.

b) Unsupervised Tagger

Brill [10] recognized that a “weakness of this rule-based tagger is that no unsupervised training algorithm has been presented for learning rules automatically without a manually annotated corpus.” So he continued his work on POS tagging and developed a tagger that is not dependent on such a training set. This new tagger, instead of relying on a manually tagged training corpus to create a lexicon, is now based solely on a pre-defined lexicon. Using this technique, each word in the unannotated text is tagged with the set of all possible tags that the word could have, i.e. the word’s lexicon entry. The tagger then learns transformations, in a fashion similar to the supervised tagger, resulting in rules generated to identify the parts-of-speech.

c) Brill Tagger Pitfall

The advantage of this type of tagger is that the algorithm for learning rules is language independent. Unfortunately, the deficiency of both the supervised and unsupervised Brill tagger is the dependency on some form of language specific resources. In the supervised case, a manually annotated corpus is needed; while the language is not a concern, the corpus is still a requirement. In the unsupervised case, a lexicon is needed, again with the language not being a concern. Many languages have some form of a lexicon available, however, not all do. More importantly still, is the case identified by the AutoTutor [24] team – a lexicon is not available for handling text that is malformed and poorly phrased. Brill’s tagger has been successfully used for tagging Swedish [28], but it was dependent on either a tagged version

Figure 4: Brill's description of transformation-based rule learning [8].
of Swedish text, or a Swedish dictionary to derive the lexicon. For the case of English tagging, Brill used a lexicon that was derived from the work done on the Penn-TreeBank Project. Neither approach is capable of handling obscure or misused words, and both approaches show the tagger’s dependence on language specific data.

2) Statistical POS Taggers

Competing with Brill’s rule-based tagger are various statistical (or stochastic) POS taggers. Linda Van Guilder [34] puts statistical POS taggers into this perspective, “the term ‘stochastic tagger’ can refer to any number of different approaches to the problem of POS tagging. Any model which somehow incorporates frequency or probability, i.e. statistics, may be properly labeled as stochastic.” Thus, one of the first principles that come to mind when discussing statistical POS taggers is Zipf’s Law [18].

a) Zipf’s Law

Li [18] summarized Zipf’s observations on word frequencies, and then proved why Zipf’s law does not significantly contribute to language understanding. “Zipf observed...that the distribution of word frequencies in English, if the words are aligned according to their ranks, is an inverse power law with the exponent very close to 1. In other words, if the most frequently occurring word appears in the text with the frequency P(1), the next most frequently occurring word has the frequency P(2), and the rank-r word has the frequency P(r), the frequency distribution is P(r) = C/r^α, with C ≈ 1 and α ≈ 1. This distribution, also called Zipf’s law, has been checked for accuracy for the standard corpus of the present-day English with very good results.”

In essence, Zipf’s law says that the second most frequent word in a corpus appears half as many times as the most frequent word, while the third appears a third as many times, and so on. This provides an interesting phenomena in language, but Li [18] points out that “probably few people pay attention to a comment by Miller in his preface to Zipf’s book, that randomly generated texts, which are perhaps the least interesting sequences and unrelated in any other scaling behaviors, also exhibit Zipf’s law. What he said was that Zipf’s law is not exclusive for English or any other natural languages.”

Because Li [18] was able to prove that Zipf’s law applies more generally to any text, he concluded that “Zipf’s law is not a deep law in natural language as one might first have thought. It is very much related to the particular representation one chooses, i.e., rank as the independent variable.” Li’s conclusion that Zipf’s law is shallow resulted from his proof that the law is a general phenomena. This generality is used later as a basis for the proposed algorithm.

b) MXPOST – N-Gram Approach

In this section, Van Guilder [34] describes the simplistic approach to statistical POS tagging, she then addresses the n-gram approach which is reinforced by Megyesi’s [21] description of the MXPOST tagger. “The simplest stochastic taggers disambiguate words based solely on the probability that a word occurs with a particular tag. In other words, the tag encountered most frequently in the training set is the one assigned to an ambiguous instance of that word” [34]. In this case, if the word “slow” is to be tagged, its most common usage would be assigned to it, i.e. JJ – adjective.

“An alternative to the word frequency approach is to calculate the probability of a given sequence of tags...this is sometimes referred to as the n-gram approach, referring to the fact that the best tag for a given word is determined by the probability that it occurs with the n previous tags” [34]. In this case, the tags play the primary function, thus the word “slow” is marked as either a verb or an adjective depending on the probability of the tags around it that indicate its part-of-speech.

“MXPOST...is a probabilistic classification-based approach based on a maximum entropy (ME) model where contextual information is represented as binary features that are simultaneously used in order to predict the POS tags. The binary features used by ME as default include the current word, the following and preceding two words and the preceding two tags. For rare and unknown words the first and last four characters are included in the features, as well as information about whether the word contains uppercase characters, hyphens or numbers. The tagger uses a beam search in order to find the most probable sequence of tags. The tag sequence with the highest probability is chosen” [21]. MXPOST uses the n-gram approach to determine the best fitting sequence of tags, however, notice that it addresses unknown words. These unknown words are those that did not appear in the training corpus, or in the lexicon. Special mechanisms exist to determine the correct part-of-speech for these unknown words.

c) TNT – HMM Approach

A mutation of the n-gram approach has produced the Hidden Markov Model approach. Thede [33] describes this approach, “in an HMM tagger the Markov assumption is made so that the current word depends only on the current tag, and the current tag depends only on the previous tag.” Thede did not mention which POS tagger was used to conduct his experiments, but the evidence seems to point at Brants’ TNT tagger.

Megyesi [21] summarizes TNT, “Trigrams’n’Tags (TNT) is a statistical approach, based on a hidden Markov model and uses the Viterbi algorithm with beam search for fast processing. The states represent tags and the transition probabilities depend on pairs of tags. The system uses maximum likelihood probabilities derived from the relative frequencies...the system uses a context of three tags. Unknown words are handled by suffix analysis, i.e. up to the last ten letters of the word. Additionally, information about capitalization is included as default” [21]. TNT was developed by Brants [7], who described his tagger as a “very efficient statistical part-of-speech tagger that is trainable on
different languages and virtually any tagset. The component for parameter generation trains on tagged corpora. The system incorporates several methods of smoothing and handling unknown words. TNT is not optimized for a particular language. Instead, it is optimized for training on a large variety of corpora. Adapting the tagger to a new language, new domain, or new tagset is very easy...Unknown words are handled by a suffix trie and successive abstraction” [7].

Megyesi [21] and Brants [7] both provided complimentary descriptions of the TNT tagger, portraying it as a robust system capable of various configurations that is fast and easy to use. It is important to note that TNT, just like the other taggers, has been developed to be language independent, provided that some form of training data or lexicon is available in the language needing to be processed. A second important note is that Brants [7] also identified the need to handle unknown words, and used a suffix trie as the solution.

A trie is a representation form that captures word characteristics in a single data structure. Cucerzan [13] used the concept of a trie in his system, and described their usefulness as able to “provide an effective, efficient and flexible data structure for storing both contextual and morphological patterns and statistics.”

Figure 5 is a graphical depiction of a trie, provided by Cucerzan [13].

![Figure 5: Cucerzan’s example of a prefix trie for “Alex and Anda are a nice couple” [13]](image)

The trie can group terms that contain similar characteristics together, allowing the system to assign a part-of-speech to an unknown word by finding the probability of known words having the same suffix as the unknown word.

d) BNC – Template Approach

Still another statistical POS tagger is the CLAWS4 system that was used to annotate the British National Corpus (BNC). However, instead of focusing on the CLAWS4 tagger, the focus is placed on the steps taken in addition to the statistical tagger to achieve the highest possible score of 99% [4]. Tagging of the BNC went through 6 steps:

1. Tokenization
2. Initial tag assignment
3. 95-96% Tag selection (disambiguation)

Steps 1-3 were handled by the CLAWS4 tagger, and the result of the tagger was an accuracy in the mid-90th percentile. Since the BNC was to be used as a standard corpus for others to use as a baseline of comparison, an accuracy of 96% was not good enough, therefore the next three steps were applied to raise the accuracy to an acceptable 99% [4].
4. 97% Idiomtagging – template matching for combining words into multi-words [4].

The idiom-tagging phase allowed the system to resolve errors due to multi-words. Multi-words include terms such as “hot dog” or other combinations of words that form a single concept, with a single part-of-speech. By correctly addressing the multi-words an additional 1% can be added to the accuracy.

5. 98% Template Tagger – apply templates to the text after multi-words have been identified to further reduce errors introduced by the multi-words, special dependencies, and long distance relationships that the tagger may have missed [4].

Once the idiom-tagging (step 4) was performed, a second template was applied to the entire corpus to see if any of the changes made by the idiom-tagging affected the resulting tagging. Additionally, special dependencies on the multi-words as well as long distance relationships between any of the words were detected during this phase. Correcting the multi-words, applying the special dependencies, and identifying and correcting long distance relationships provided another 1% accuracy.

6. 99% Postprocessing: including Ambiguity tagging – allowing certain words to have ambiguous tags marking the word as two categories [4].

The final step in the BNC tagging process was a postprocessing stage, which involved a final check through the corpus to determine which words were truly ambiguous. For the words that could not be determined to have a single part-of-speech an “ambiguous tag” was permitted to exist in the corpus. This final process of allowing ambiguous tags provided the additional 1% accuracy needed to achieve 99%.

3) Autotutor – Neural Network Approach

The last POS system presented is AutoTutor, the system mentioned in previous sections. Olde et. al [24] describe the mission of their system, “AutoTutor is a fully automated tutoring system that attempts to comprehend learner contributions and formulate appropriate dialog moves.” The first step in creating this system was language understanding, which required a POS tagger. Since the AutoTutor team felt their application was unique, they developed their own POS tagger. Their tagger operates similarly to the others presented in this section, “it first consults a lexicon to identify the set of possible tags for each word, then uses a neural network to select a single tag for each word” [24].

The similarity between this system and the others presented is its dependence on a lexicon as the first step in its POS tagging efforts. While other systems have been designed to be language independent, AutoTutor [24] may have a slight advantage over them because it was designed with an additional requirement to handle input that is malformed and full of errors.

AutoTutor “uses a neural network to incorporate surrounding contextual cues to determine the single most likely POS tag” [24]. The neural network “uses local context (the words preceding and following the target word and its position in the sentence) and base rate frequency information to select the most likely POS tag from the set of candidate tags handed up by the lexicon” [24].

4) Approach Commonality

While the algorithm for assigning the parts-of-speech may be different, the approach taken by each system is similar to all the others because they rely on a lexicon as the basis for choosing an answer. A lexicon often times does not contain entries for some of the words appearing in the corpus, leaving so called “unknown words” to be disambiguated by the POS tagger. All the systems have mechanisms that differ greatly for dealing with unknown words, even though they use similar information, i.e. context, morphology, and capitalization.

5) Intro to Unknown Word Guessing

The work done by Nakagawa [23] is used to introduce the problem of handling unknown words, “in part-of-speech tagging, we frequently encounter words that do not exist in training data. Such unknown words are usually handled by an exceptional processing, because the statistical information or rules for those words are unknown…though these methods have good performance, the accuracy for unknown words is much lower than for known words, and this is a non-negligible problem…”

D. Unknown Word Guessing

There are two classes of words, they are open and closed class. The “closed class consists of a finite and well-established list of words such as prepositions, articles, wh-words, etc.” [22]. The open class consists of all the other words that can be used in language; these include nouns and verbs, that essentially have an infinite number of members. Unknown word guessing relates only to open class words since assigning their part-of-speech is likely to be ambiguous.

As described in the previous section, POS taggers rely on lexicons to perform their intended function. Unfortunately, there is a disadvantage to this approach – the disambiguation of unknown words must be handled by a separate mechanism. Since the lexicon is created using a limited data set, it does not contain all the possible words the tagger will see in future texts, thus, creating the need for the POS tagger to handle such unknown words. The work done by Mikheev [22] shows there is interest in making lexicons more robust by attempting to create an entry for these unknown words. Prior to Mikheev, the typical way of resolving an unknown word was to assign it a single tag; this is contrasted with Mikheev’s method of assigning a class of tags. By assigning a class of tags to a word, the POS disambiguator can be used to assign the part-of-speech. This approach provides the ability to produce a more accurate result, while the alternative (assigning a single tag) bypasses the sophisticated
disambiguation mechanisms in the POS tagger, potentially leading to poorer results.

1) Context and Morphology

Orphanos [25] described a common method for handling unknown words in POS taggers by saying, “in order to increase their robustness, most POS taggers include a guesser, which tries to extract the POS of words not present in the lexicon. As a common strategy, POS guessers examine the endings of unknown words along with their capitalization, or consider the distribution of unknown words over specific parts-of-speech. More sophisticated guessers further examine the prefixes of unknown words and the categories of contextual tokens” [25]. The notion of using context and word characteristics (i.e. morphology) has proven to be the standard practice in assigning roles to unknown words.

Thede [33] stated, “one common approach [to handle unknown words] is to use affixation rules to “learn” the probabilities for words based on their suffixes and prefixes.” Nakagawa [23] reinforced this idea by saying, “one known approach for unknown word guessing is to use suffixes or surrounding context of unknown words.” Vasilakopoulos [35] agrees that using context and morphology is the proper way to handle unknown words saying, “an appropriate and effective guesser should check the morphology of the current word, as well as the context (the previous and next tokens).” Finally, Mikheev [22] described a more advanced approach that uses leading and trailing word segments to achieve an accuracy of up to 85% on unknown words.

In contrast, Van Guilder [34] declared that context and morphology are not the only aspects of the text available to assign parts-of-speech. She said, “some systems go beyond using contextual and morphological information by including rules pertaining to such factors as capitalization and punctuation. Information of this type is of greater or lesser value depending on the language being tagged” [34]. Van Guilder [34] explained that “in German…information about capitalization proves extremely useful in the tagging of unknown nouns.” It’s the characteristics of a particular language that allow POS taggers to be customized for better performance. All unknown word guessers use some type of rule set to determine the correct part-of-speech. Thede [33] has provided one such example for English, “a word ending in –ed is likely to be a past tense verb or a past participle.” Another example is the set of rules used to tag the British National Corpus [4]:

- Look for the ending of a word: e.g. words in -ness will normally be nouns.
- Look for an initial capital letter (especially when the word is not sentence-initial). Rare names which are not in the lexicon and do not match other procedures will normally be recognized as proper nouns on the basis of the initial capital.
- Look for a final -(e)s. This is stripped off, to see if the word otherwise matches a noun or verb; if it does, the word in -s is tagged as a plural noun or a singular present-tense verb.
- Numbers and formulae (e.g. 271, *K9, β+) are tagged by special rules.
- If all else fails, a word is tagged ambiguously as either a noun, an adjective or a lexical verb [4].

These rules provide an introduction into the methods presented by Nakagawa [23], Mikheev [22], Thede [33], and Cucerzan [13]. The common thread between all the approaches is their use of context coupled with language dependent rules for analyzing prefixes and suffixes. Even though the POS taggers presented above claim to be language independent, they require language specific rules that leverage prefix and suffix information to correctly annotate unknown words.

2) Unknown Word Guessing Methods

In this section, three methods for unknown word guessing are presented: support vector machines, decision tree induction, and statistical methods. The first two methods are presented very briefly to show that there are multiple ways of attacking unknown word guessing. These methods are capable of achieving respectable results and are versatile algorithms that can be used in numerous domains.

The first method, called a support vector machine, has been demonstrated by Nakagawa [23]. “Support vector machines are a supervised machine learning algorithm for binary classification and known to have good generalization performance” [23]. This algorithm was designed to measure the distance between two vectors, each representing a word and its characteristics, to determine the likelihood that the word falls into a certain category.

Nakagawa [23] used a context that included the two preceding tags along with two preceding and succeeding words. He also used prefixes and suffixes of up to four characters. Using a training corpus of 1,000 tokens the system was capable of achieving a score of 64.2% and with a training corpus of 1,000,000 tokens the score rose to 80.8%. Nakagawa [23] emphasized the importance of affixes to correctly identify the POS tag, without affixes the scores for the respective corpus sizes reduce to 33.7% and 30.0%.

The second method, called decision tree induction, has been demonstrated to be a versatile algorithm. Vasilakopoulos [35] has used Decision Tree Induction for a variety of tasks, and claimed that the exact same system has been used for unknown word guessing, text chunking, and named entity recognition.

The third method, the statistical approach, is discussed in much more detail below referencing Mikheev [22], Thede [33], and Cucerzan’s [13] algorithms.

a) Mikheev’s Approach

Instead of assigning a single part-of-speech to an unknown word, Mikheev’s approach [22] differs from the others because it attempts to create an entry in the lexicon that can be used by the POS tagger to disambiguate the unknown
words. His approach is meant to be a way of improving the lexicon, therefore requiring an existing lexicon. The algorithm removes affixes (prefixes and suffixes) from unknown words and then compares the result to the existing lexicon. If an existing term in the lexicon matches the affix-removed word, then a class of tags is assigned to the word and the entry is added to the lexicon. The assigned class is determined by the affix and the existing lexicon entry. Once a class has been assigned, the tagger’s disambiguator can run using the new lexicon.

(1) Framing the Problem

Mikheev has framed the problem in the following way, “words unknown to the lexicon present a substantial problem to NLP modules (as, for instance, part-of-speech (POS-) taggers) that rely on information about words, such as their part of speech, number, gender, or case. Taggers assign a single POS-tag to a word-token, provided that it is known which POS-tags this word can take on in general and the context in which this word was used. A POS-tag stands for a unique set of morpho-syntactic features, [as described in the table below], and a word can take several POS-tags, which constitute an ambiguity class or POS-class for this word. Words with their POS-classes are usually kept in a lexicon. For every input word-token, the tagger accesses the lexicon, determines possible POS-tags this word can take on, and then chooses the most appropriate one. However, some domain-specific words or infrequently used morphological variants of general-purpose words can be missing from the lexicon and thus, their POS-classes should be guessed by the system and only then sent to the disambiguation module” [22]. The table below lists some of the most frequent open class tags from the Penn-TreeBank tag set. Mikheev [22] has shown that assigning a part-of-speech tag to an open class word can be done through the use of morphology.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Meaning</th>
<th>Example</th>
<th>Tag</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS</td>
<td>common noun</td>
<td>table</td>
<td>VB</td>
<td>verb base form</td>
<td>take</td>
</tr>
<tr>
<td>NNS</td>
<td>noun plural</td>
<td>tables</td>
<td>VBD</td>
<td>verb past</td>
<td>took</td>
</tr>
<tr>
<td>NNP</td>
<td>proper noun</td>
<td>John</td>
<td>VBG</td>
<td>gerund</td>
<td>taking</td>
</tr>
<tr>
<td>NNSP</td>
<td>plural proper noun</td>
<td>Vikings</td>
<td>VBN</td>
<td>past participle</td>
<td>taken</td>
</tr>
<tr>
<td>JJ</td>
<td>adjective</td>
<td>green</td>
<td>VBD</td>
<td>verb present, 3d person</td>
<td>takes</td>
</tr>
<tr>
<td>RB</td>
<td>adverb</td>
<td>naturally</td>
<td>VBP</td>
<td>verb, present, non-3d</td>
<td>take</td>
</tr>
</tbody>
</table>

Table 1: Open-class tags [22]

(2) Naïve and Simple Approaches

Mikheev [22] described the naïve approach to assigning POS tags to unknown words by saying, “the simplest approach to POS-class guessing is either to assign all possible tags to an unknown word or to assign the most probable one, which is proper singular noun for capitalized words and common singular noun otherwise. The appealing feature of these approaches is their extreme simplicity. Not surprisingly, their performance is quite poor.” He has suggested an improvement on this naïve approach claiming “a simple probabilistic approach to unknown-word guessing...is more accurate than the naïve assignments and easily trainable, [however] the tagging performance on unknown words is reported to be only about 66% correct for English” [22].

Mikheev [22] based his algorithm on the principle that, “unlike many other approaches, which implicitly or explicitly assume that surface manifestations of morpho-syntactic features of unknown words are different from those of general language, we argue that within the same language unknown words obey general morphological regularities.” To supplement this principle, a more practical definition was provided, “in English, as in many other languages, morphological word formation is realized by affixation: prefixation and suffixation” [22]. By combining these two concepts, Mikheev has emphasized that unknown words within a language obey general morphological rules that can be detected by analyzing the prefixes and suffixes of words.

To properly scope his claim, Mikheev treated words that appeared only once as a special case. Words appearing exactly once were called “hapax words.” Mikheev used these hapax words as his test set to see if his algorithm could correctly categorize them. The significance in retaining the hapax words was the algorithm’s ability to perform analysis of affixes, comparing them against the words contained in the lexicon. Without a lexicon to compare against, words appearing only once in a corpus are not useful. The result of his algorithm’s affix analysis was a set of rules that could be applied generally. To evaluate the rules, Mikheev removed all hapax words from the lexicon that tag the Brown Corpus using two different taggers, a stochastic tagger and a rule-based tagger [22].

To provide an example of Mikheev’s algorithm a sample rule is shown below. This example shows that the rules induced by Mikheev’s system are a transformational style, focusing on individual words. The sample rule is:

[-ied +y ?(VB VBP) → (JJ VBD VBN)]

This rule says that if there is an unknown word that ends with ies, we should strip this ending from it and append the string ies to the remainder. If this modified word is found in the lexicon as (VB VBP) (base verb or verb of present tense non-3d form), we conclude that the unknown word is of the category (JJ VBD VBN) (adjective, past verb, or participle). Using the word specified as an example, if it was unknown to the lexicon, this rule first would try to segment the required ending ies (specified → ies = specify), then add to the result the mutative segment y (specify + y = specify) and, if the word specify was found in the lexicon as (VB VBP), the unknown word specified would be classified as (JJ VBD VBN) [22].

(3) Comparing Mikheev and Brill

Mikheev [22] concedes that his proposed guessing rules are similar to the rules developed by Brill. However, Mikheev has noted 3 major differences

• “Brill’s transformations do not check whether the stem belongs to a particular POS-class” while the
Mikheev’s approach does and therefore imposes more rigorous constraints; [22]

• “Brill’s transformations do not account for irregular morphological cases” like “cry” vs. “cries” whereas Mikheev’s approach does; [22]
• “Brill’s guessing rules produce a single most likely tag for an unknown word, whereas [Mikheev’s] guesser is intended to imitate the lexicon and produce all possible tags.” [22]

Further defining Mikheev’s approach, the following 6 types of words were handled by his guessing-rules:

- Words with no mutative endings; for example, book + ed = booked;
- Words with mutative endings; for example, try – y + ied = tried;
- Words with no mutative prefixes; for example, un + screw = unscrew;
- Words containing hyphens;
- Words containing capital letters;
- All other words. [22]

b) Thede’s Statistical Approach

Thede [33] has stated that word endings, word beginnings, and context are valuable for categorizing unknown words. However after performing some experiments, he explained that only endings are true discriminators [33]. Thede took Mikheev’s approach one step farther, instead of relying on the lexicon to determine the proper part-of-speech, the lexicon was used to create predictors that can be used independently to assign unknown words to a part-of-speech. Thede’s initial experiment led him to believe that “by totaling the tags for every word with a particular word ending and word beginning, a probability distribution can be created for the unknown word predictor” [33]. Unfortunately, these probabilities were unbalanced and caused too many misclassifications. To correct the problem Thede [33] tried to use a measure of disorder.

“Disorder is a term used in information theory. It is a measure of how much information is necessary to separate data. The average disorder of a tag distribution is found with the following equation:

\[
\text{Disorder} = \sum (n_i/N) \ln(n_i/N),
\]

where \(n_i = \) the number of occurrences of the \(i\)-th tag and \(N = \) the total number of occurrences” [33].

Thede [33] was faced with more disappointment as the measure of disorder also failed to provide sufficient results. “Even using the disorder calculations, there were still some mistags from using the word-beginning information instead of the word-ending. So, the predictor was designed to use only the word-ending information” [33]. Thus, Thede [33] concluded that only word-ending information was useful, and claimed that “this work seems to offer great promise in the use of statistics in tagging unknown words. Preliminary results are near to current best averages, without relying on language-specific information.” Thede [33] claimed no language-specific information was needed because his algorithm was capable of learning predictors. His assumption was that a lexicon is not considered language-specific information.

c) Cucerzan’s Approach

Working on a different problem, named entity recognition, Cucerzan [13] developed an algorithm that can be applied to unknown word guessing that “relies on both word internal and contextual clues.” It uses the “morphological structure of the word and makes use of the paradigm that for certain classes of entities some prefixes and suffixes are good indicators” [13]. To capture the information in the context and the morphology, Cucerzan [13] used 4 tries, one each for left and right context, and one each for prefixes and suffixes.

Cucerzan [13] justified using context to identify similar terms by saying that a newly introduced term will be repeated “if not for breaking the monotonous effect of pronoun use, then for emphasis and clarity.” Thus suggesting that new terms are used in a similar context to the known terms. “By gathering contextual information about the entity from each of its occurrences in the text and using morphological clues as well, we expect to classify entities more effectively than if they are considered in isolation” [13].

Since Cucerzan’s algorithm relies only on training data, it is only capable of performing as well as the training provides. Therefore, a condition is placed on the statement above pertaining to context identifying similar terms. This condition is placed on unique instances of words, “clearly, in many cases, the context for only one occurrence of a new word and its morphological information is not enough to make a decision” [13]. A single instance of a word does not provide sufficient information to learn the rules needed to categorize the word. But for the words that can be correctly categorized, Cucerzan [13] has provided a way to retain its information, “the newly classified tokens and contexts are saved for future use as potential seed data in subsequent…classification on new texts.”

d) Summarizing Unknown Word Guessing

Mikheev [22] has provided a very robust way of identifying unknown words, by leveraging the information contained within an existing lexicon, his algorithm is capable of analyzing the affixes of words and properly categorizing the terms. Unfortunately, Mikheev’s approach is only useful in the case where an existing lexicon is readily available and sufficiently complete. Thede [33] attempted to take a step away from direct use of the lexicon to predict unknown words by using a lexicon to calculate predictors for unknown words based on word-endings. Finally, Cucerzan [13] presented an approach that was not dependent on any lexicon, but rather only on unannotated text and some training examples. Although Cucerzan’s approach did not perform as well as Mikheev’s, by combining their efforts with the work of Brill [10], a foundation has been created in this paper to do two things: fully automated POS tagging; and running POS taggers on any language.
E. Miscellaneous

1) Clustering and Multi-Words

In the previous section, unknown words were discussed as members of the open-class set of words. Mikheev’s algorithm [22] focused on open-class words, and he went to significant effort to prevent other words from interfering with his algorithm. This is how he described his effort; “we filtered out words shorter than four characters, nonwords such as numbers or alpha-numerals, which usually are handled at the tokenization phase, and all closed-class words, which we assume will always be present in the lexicon” [22]. Here we see that some words contain characteristics that can be used to categorize them immediately and there are other words that are assumed to fit into an always known set. However, these assumptions only hold when the language being processed is well understood. Additionally, some of the closed-class words are useful for other aspects of lexicon creation. One such aspect is clustering and another is the identification of multi-words.

a) Clustering

Clustering has been considered in this paper because the ability to group similar words together can be a significant advantage in automatically generating a lexicon. Alexander Clark [12] has focused on grouping words into morphologically similar categories, and explains “we are particularly interested in rare words…it is most important to cluster the infrequent words, as we will have reliable information about the frequent words; and yet it is these words [the infrequent words] that are most difficult to cluster.” Clark is focused on rare words and how to cluster them into similar groups. He has considered two approaches: one that assigns each of the \(n-1\) most frequent words to a separate class, and all other words to another class; and a second approach that assigns every word to its own class. The second is a technique used in text-retrieval to cluster documents. This is described by Weiss [36] below.

Weiss [36] has described clusters as a way to “organize an information space for the user and the system by grouping related subspaces together. Subspaces may be clusters of documents or clusters of clusters. The partitioning of information space provides convenient abstraction barriers for both the user and the system.” With clusters the user and the system can perform better due to the similarity among the group members. Similar to Clark’s [12] second approach, which considered each word as a separate class, Weiss [36] has used an algorithm that “starts with a set where each original document represents an independent cluster. The algorithm iteratively reduces the number of clusters by merging the two most similar clusters until” a threshold has been met. Although Weiss [36] has focused on document clustering rather than word clustering, his technique and the one used by Clark have the same basic foundation – that each item-of-interest is placed into its own class, then iteratively combined with the next most similar class to create a cluster.

Let

\[
    \begin{align*}
    w_{ki}^{ij} & = \text{contribution to weight from frequency } f_{ki} \\
    w_{ki}^{ds} & = \text{contribution to weight from size of } d_k \\
    w_{ki}^{at} & = \text{contribution to weight from term: attribute}
    \end{align*}
\]

then

\[
    w_{ki}^{ij} = (0.5 + 0.5 \frac{f_{ki}}{\max_j \{f_{kj}\}})
\]

\[
    w_{ki}^{ds} = \frac{1}{\sqrt{\sum_i (w_{ki}^{at}w_{ki}^{ds})^2}}
\]

\[
    w_{ki} = w_{ki}^{ij} \cdot w_{ki}^{ds} \cdot w_{ki}^{at}
\]

Figure 6: Term Weighting Equation [36]

“The term-based similarity function \(S_i\) between documents \(d_i\) and \(d_j\) is the normalized dot product of the terms vectors representing each document” [36]. Vectors representing the words contained in a document can be used to compare the similarity of two documents, and if the similarity falls within a predefined threshold then the documents are clustered. This method was considered for the lexicon generation algorithm presented later in this paper, but was not used because the calculation for term similarity is dependent on \(w_{ki}^{at}\), a weight assigned to some attribute of the term indicating its significance in the document. Traditionally, this weighting comes from the location of the word in the document, for example giving a higher weight to a word appearing in the title of a document. However, when all words are considered the same, then no word is more significant than another, and the weighting is equal for all words in the document. This approach may be worth revisiting if a significance factor for each word can be determined directly from other characteristics in the text.

b) Multiwords

Another important aspect to consider when evaluating the use or need of closed-class words is the identification of multi-words. Zhai [37] pointed out that “one important aspect of acquiring a lexicon is the acquisition of lexical
atoms...a good example is hot dog, where the meaning of the whole phrase has almost nothing to do with the literal meaning of hot or dog.” Closed-class words often make up parts of multi-words, and if they are ignored or disregarded then the accuracy provided by identifying multi-words could be lost. As shown by the BNC [4], the identification of multi-words is critical in obtaining the 99% accuracy needed for creating a standardized corpus (i.e. a corpus that will be used commonly as the basis for future research).

There are different types of multi-words, such as multi-word nouns (i.e. “hot dog”) and multi-word verbs (i.e. “bog down”). Blaheta [3] has proposed a method for identifying multi-word verbs in an unsupervised fashion using existing part-of-speech tags and a calculation of “log-linear models.” Here are some of the top multi-word verbs from his experiments:

- consist of
- fend off
- shy away
- bog down
- beef up
- bail out
- lag behind
- make up of
- miss out on
- own up to
- crack down on

Blaheta’s examples provide a clear picture of the types of words that should be considered together as a single multi-word, rather than as individual words. For a more formal definition we look to Thanopoulos [32] who has summarized the work done by Cruse. “Cruse, defining the notion of the textual entity word from the perspective of contextual lexical semantics, describes it as “the lexical element which is typically the smallest element of a sentence which has positional mobility and the largest unit which resists interruption by the insertion of new material between its constituent parts.” Although the word is indeed the lexical element which typically complies with both requirements, there are plenty of word sequences which satisfy them as well, such as New York and kick the bucket. Cruse describes them as “minimal semantic constituents which consist of more than one word”; we call them non-compositional multi-words” [32].

Unlike Blaheta [3], Thanopoulos [32] did not rely on POS tags to perform his multi-word identification. He mentioned, “since existing electronic lexico-semantic resources, e.g. Wordnet, lack full coverage...automatic acquisition of such knowledge from corresponding text corpora is an attractive and economic solution.” Thanopoulos [32] relied heavily on context to calculate a likelihood ratio that can identify two words that should be a multi-word. “Approaches using large context windows are computationally expensive and their output indicates topic-similarity rather than semantic similarity...having confirmed this by comparative experiments, we employed only next and previous word adjacency” [32].

Similarly, Cucerzan [14], in an attempt to correctly categorize gender-marked words, selected a context window size of ±3 words. He claimed that “beyond this window the agreement/disagreement ratio approaches chance.” He also claimed that a smaller window trades lower coverage for increased accuracy [14]. Despite Thanopoulos [32] and Cucerzan’s [14] approaches using a small context window, Zhai [37] used a context window of 50 words.

Although it is rare to use a context window as large as the one described by Zhai [37], a broader use of the resulting multi-words has been identified; “because the meaning of lexical atoms is non-compositional, naturally, they must be recognized and treated as a single unit...in information retrieval, it is desirable to recognize and use lexical atoms for indexing. In machine translation, a lexical atom needs to be translated as a single unit, rather than word by word” [37].

Even though the context window size varies among algorithms, their goal and output remain similar. Since many of the POS taggers described earlier use a context window of one to four words/tags, Zhai’s algorithm is being considered an outlier.

As shown above, the concept of identifying multi-words is important and useful for refining the accuracy of a lexicon. It is also a valuable area that should be considered for future work in lexicon generation. However, identifying multi-words is not part of the algorithm presented in this paper because it is considered a refinement of the more general approach – lexicon generation. The algorithms for identifying multi-words were a useful starting point for the lexicon generation algorithm because they incorporate aspects of, and work in conjunction with, clustering to determine how similar words can be grouped into their proper POS categories.

2) Measurements

Once an algorithm for assigning parts-of-speech has been executed, and once the unknown and multi-words have been identified, then the result needs to be measured in some way. There are two ways POS tagging can be evaluated, precision/recall or accuracy. In the precision/recall approach, precision is the percentage of POS-tags the tagger assigned correctly divided by the total number of POS-tags it assigned to the word; recall is the percentage of POS-tags correctly assigned by the guesser. Mikheev [22] appropriately noted, “precision seems to be slightly less important since the disambiguator should be able to handle additional noise but obviously not in large amounts.”

The second way to measure the result of a POS tagger is described by Megyesi [21], “evaluation is based on the widely used measure, accuracy, which is obtained by dividing the number of correctly labeled tokens with the total number of tokens.” This definition of accuracy is shown below:

\[
\text{Accuracy} = \frac{\text{No. of correctly tagged tokens}}{\text{Total no. of tokens}}
\]

Figure 7: Defining Accuracy [21]
Accuracy has been selected as the evaluation method for the lexicon generation algorithm presented in this paper.

3) Comparison Fairness

In developing their algorithm for language independent named entity recognition, Cucerzan [13] noted that “it would be inappropriate to compare the results of a language independent system with the ones designed for only one language. As Day and Palmer observed, “the fact that existing systems perform extremely well on mixed-case English newswire corpora is certainly related to the years of research and organized evaluations on this specific task in this language. It is not clear what resources are required to adapt systems to new languages.”” Cucerzan [13] reinforced this argument by saying, “what should be underlined here is that these systems were trained for a specific domain and a particular language, typically making use of hand-coded rules, taggers, parsers and semantic lexicons.” This same measure of comparison can be applied to the automatic lexicon generation approach described in this paper. Since the goal is to create an algorithm that is not dependent on any outside information, the same conditions apply, making it inappropriate to compare the result of this algorithm with others using specialized information for a particular language.

4) Scores for Existing Systems

a) POS Tagging Scores

In order to provide some basis for comparison, measurements reported on the methods described earlier for POS tagging, unknown word guessing, and named entity recognition are presented here.

In terms of part-of-speech tagging, the BNC used a set of sophisticated methods to achieve the highest possible score, 99% [4]. While this is the goal for all POS taggers, the BNC used more than just a POS tagger to achieve its results. To summarize the prior section, Brants [7] defines accuracy as “the percentage of correctly assigned tags.” Given such an accuracy measurement, Brants makes a broad statement regarding the current state of POS taggers on standardized corpora, “average accuracy on unseen English text from the same domains as the Susanne corpus is around 96%” [7]. He has provided a specific measurement, made on his POS tagger on a specific corpus, saying that TNT is capable of achieving 96.7% accuracy on the Wall Street Journal (WSJ) corpus from the Penn-TreeBank [7].

Megyesi [21] has performed a survey of several POS taggers and has concluded that the “best performance can be obtained by training on the basis of POS tags with labels marking the phrasal constituents without considering the words themselves.” Megyesi verified that POS taggers produce the best results when given training data containing POS tags. But more importantly, Megyesi said that POS taggers learn sentence structure, and given POS tags as training, the words become less important. Unfortunately, when there are no POS tags, and only the words are available, the sentence structure cannot be determined without knowing which words fall into which categories; emphasizing further the need for a lexicon. When Megyesi [21] “extended the templates of Brill’s POS tagger to include references up to two chunk tags…they achieved 88% for partitioning the sentence into N and V chunks when trained on 200k words.” Here Megyesi has shown that even separating nouns from verbs is not a trivial problem.

While comparing three POS taggers, Megyesi [21] ran an experiment using a corpus consisting of two hundred thousand words and provided a lexicon to the taggers, but no training was provided showing how the parts-of-speech were assigned to the words in the corpus. As a result, the accuracy of the taggers varied from 67.59% to 77.86%, and is shown in Table 2.

The accuracy of each parser trained on 200k tokens is shown

Table 2: POS Tagger Accuracy [21]

b) Unknown Word Guessing Scores

Given the diversity of algorithms and outside sources of information, the tagging accuracy of unknown words varies greatly. Nakagawa [22] said “the tagging performance on unknown words is reported to be only about 66% correct for English.” However, Thede [33] achieved an accuracy score of 70.9% using context and word-endings, while Brants [7] claimed that TNT is capable of achieving 79.5% for unknown words on the WSJ corpus. This is augmented by Vasilakopoulos’ [35] report claiming the system he developed was able to achieve accuracy between 85% and 87% when applied to unknown words. Finally, Cucerzan [13] was able to achieve an F-measure, which is the combination of precision and recall, of 70.5%-75.4% for correctly categorizing named entities, such as person names or locations. With such a vast difference in reported results, and the dependence of each algorithm on various language specific data, the range for categorizing unknown words into some type of category (i.e. parts-of-speech or named entities) lies between 65% and 85%.

The diversity of algorithms and outside sources of information, the tagging accuracy of unknown words varies greatly. Nakagawa [22] said “the tagging performance on unknown words is reported to be only about 66% correct for English.” However, Thede [33] achieved an accuracy score of 70.9% using context and word-endings, while Brants [7] claimed that TNT is capable of achieving 79.5% for unknown words on the WSJ corpus. This is augmented by Vasilakopoulos’ [35] report claiming the system he developed was able to achieve accuracy between 85% and 87% when applied to unknown words. Finally, Cucerzan [13] was able to achieve an F-measure, which is the combination of precision and recall, of 70.5%-75.4% for correctly categorizing named entities, such as person names or locations. With such a vast difference in reported results, and the dependence of each algorithm on various language specific data, the range for categorizing unknown words into some type of category (i.e. parts-of-speech or named entities) lies between 65% and 85%.
5) Experiments by Others

Part-of-speech taggers have been the focus of numerous experiments, including some that have been conducted to measure the effectiveness of their various parts. These measurements include the effect of lexicon size, the number of unknown tokens and how to resolve them, comparing different types of taggers, and identifying parts-of-speech that are difficult to distinguish between. The first experiment was conducted by Vasilakopoulos [35] and focused on lexicon size. He reported, “if we double the lexicon size, we can achieve better results than…with half the lexicon size, even if the training corpus size is significantly smaller” [35]. His experiments were used to show that increasing the size of the lexicon could be used to produce better results than reducing the size of the corpus.

A second set of POS tagger experiments was conducted by Megyesi [21], and the results highlighted in this section focus on systems trained “on the basis of the word only – lexical information – to predict the POS tag.” Megyesi [21] described, “the first experiment, where training is performed on the basis of lexical information only to predict the POS together with the correct phrase labels (WORD → POS & PHRASES), is the hardest classification task for every algorithm, see figure…This is not surprising since the algorithms have to learn a great number of classes…thus, in this experiment the hypothesis space that the algorithms have to search through is large. The classifiers here are treated as POS taggers and parsers. TNT has the lowest error rate on small training data while MXPOST outperforms TNT when using above 50k tokens for training. However, fnTBL [Brill] achieves higher performance than MXPOST on small data sets.” The result of the experiment is shown in Figure 8. Megyesi [21] showed that all the taggers perform similarly even though each one incorporates a different algorithm. Additionally, the taggers performed at different accuracies based on the size of the training data.

![Figure 8: Comparing POS taggers using word-only to predict parts-of-speech [21]](image)

The size of the corpus makes a significant difference to the performance of the tagger because “the number of the unknown tokens when using small training data is high (51% for 1k tokens, and 20% for 50k tokens, respectively), [showing] why good morphological analysis is needed.” Megyesi [21] demonstrated that unknown words are common, but as the corpus size increases, it is expected that more of the known words will be reused, thus reducing the total number of unknown words in the corpus. In any case, but especially when provided little training, Megyesi [21] considered morphology to be the distinguishing factor in resolving the unknown words, saying “the success of TNT in the POS and phrase label assignment…depends on the parameters used for the annotation of unknown words…TNT checks up to the last ten character of a token while the other approaches use suffix analysis up to four characters only.”

When considering unknown words, Megyesi’s experiments conclude that Brants (TNT) performed well due to its advanced suffix analysis, while Brill performed the worst on small data sets. However, the Brill tagger outperformed the other systems overall due to its larger context window of three preceding and following words and tags [21].

The final experiment presented in this section was conducted by Orphanos on his native language, Greek. He has identified combinations of part-of-speech categories that are difficult to differentiate, both with his algorithm and with a more general approach. His results are used as a basis for comparison later in this paper for the lexicon generation algorithm. The rightmost column in Table 3 shows that even with a trainable method for resolving unknown words (decision trees), distinguishing between adjectives and adverbs is the most problematic, producing double-digit error rates when considered as a decomposition of the aggregate error rate. The dark shaded rows in the table indicate the four highest error rates for resolving ambiguity between parts-of-speech categories. Fortunately, these ambiguities represent such a small percentage of the corpus (% occurrence in the corpus) that the total error rate is still within a reasonable bound [25].

![Table 3: POS Ambiguity Schemes – Showing adjectives and adverbs have high error rates [25]](image)
III. Method

A. Need for Automatic Lexicon Generation

The need, use, and dependence on lexicons has been demonstrated in previous sections, however, the need for automatically generating one has not been emphasized enough. The Penn-TreeBank was used as an example to show the effort involved in creating a lexicon. Additionally, the story by Zinsser [38] described the effort in creating a dictionary. The AutoTutor system [24] was used to show that there are applications that require POS tagging beyond written text that contain words found in a traditional dictionary or lexicon. In order to handle these cases, a new approach must be devised. I propose such a method, a framework that can be used to generate a lexicon solely from an unannotated corpus.

1) Breaking Assumptions

There are two assumptions that must be broken in order to justify a truly unsupervised language independent POS tagger. The first and most important assumption that needs to be broken is the acceptance that existing lexicons are available for every language (or type of language). While the second assumption that needs to be broken is that algorithms are language independent even when they rely on existing lexicons, which were created for a certain subset of a language(s).

One of Brill’s colleagues has illustrated the assumption that a lexicon is common knowledge and does not fall under the definition of an unsupervised tagger. While discussing Brill’s algorithm, his colleague said, “…during our tests we used a lexicon derived from the training and the test data. This approach violates the assumption of “unsupervised training.” Tests need to be performed using a generic lexicon such as one obtained from the Webster dictionary” [27]. Clearly the lexicon is being taken for granted, and in my opinion, even using a lexicon based on a dictionary still does not define a truly unsupervised system. As we saw from Zinsser’s [38] discussion at the beginning of this paper, many hours of debate go into deciding if a word, and its usage, will enter the dictionary. A true unsupervised language-independent tagger would be able to create a lexicon by starting with an untagged corpus of any language, identify the categories that words can fall into, and then proceed with the rest of the tagging algorithm.

2) Usefulness

A useful application of an automatically generated lexicon is in handling the improper use of a language. If a corpus full of improper usage and misspellings was presented to a lexicon-dependent system, the system would perform poorly because the number of unknown words would be very high. Even if a dictionary were available for every language, nothing can prevent people from using a language incorrectly. There is also no way to prevent new words from being created, or existing words from being misspelled. Handling misuse of a dictionary would require the dictionary to contain every possible usage of a word, not just the correct uses. Additionally, a dictionary would need to contain every possible misspelling. With each new use or spelling variation of a word, the dictionary and lexicon entry would need to be updated. However, if the system were designed to treat every word as an unknown word from the beginning, and create a lexicon for each corpus shown to it, then misuse and misspellings would no longer be a concern.

Brill [8] has provided an example of a mutated case:

Consider the following experiment. We take equal portions of French and English text, and then make a new text by repeatedly moving one word picked randomly from either text to the new text. Next we give the text to somebody who knows neither English nor French and ask them to take each word appearing in the mixed up text, and label the word as either being French or English. If the person picked randomly, they would be 50% correct. If we were to provide the person with a list of the 10 most probable words in both English and French, an accuracy of 63% would be obtained. If the word list was extended to 50 words, 71% accuracy would be possible. If instead the person was asked to build a dictionary listing which words appearing in the text are English and which are French, and accuracy based upon the percentage of correct dictionary entries, then assuming a text size of one million words, giving the two lists of 50 words would give an accuracy of only 50%.

Here a distinction has been made between identifying words in a corpus and identifying words in a dictionary. Since words in a corpus are often reused to convey meaning, it is more likely that a higher percentage of words can be identified simply because of their reuse. If a word is used in a certain way one time, there is a very high probability that it will be used again the same way.

Brill’s case of identifying language is an easier problem than POS tagging because it is a binary choice, once a word is identified as being from a given language, its categorization will never change. Perhaps the same word can be used in more than one language, but that does not remove it from being a member of the other language. Thus, being given 50 words in a corpus has a much higher significance than being given 50 words from a dictionary; simply because of the redundancy found in any given corpus. Brill [8] said the usefulness in finding the meaning of only a few words can significantly increase tagging accuracy—“although only a small percentage of words that appear in a corpus appear with a high frequency, those high frequency words account for a large percentage of the total tokens in the corpus.”

3) Languages

Related work by Abney [1] showed that specialized taggers have been written for various languages including Basque, Dutch, French, German, Greek, Italian, Spanish, Swedish, and Turkish. However, these taggers are independent of each other, and none of them are general enough to handle all the languages.
In an attempt to use the Brill tagger on another language, Prütz [28] tried it on Swedish with unsatisfactory results. He explained, “originally the tagger was trained using the method described by Brill (1992). The result was not satisfactory. Only some 89% of the tokens in the text corpus were correctly tagged (91.5% using the limited tagset)” [28]. Unfortunately, the result of the tagging was much lower than he anticipated. He was probably expecting the 96.5% accuracy Brill reports on his English studies [9].

With further investigation, Prütz [28] identified a significant problem using Brill’s tagger on Swedish – he found words in his corpus that were not in the lexicon, perhaps because they were misspelled, or perhaps because he didn’t have a Swedish lexicon of a size sufficient to handle his corpus. This is how he described the situation: “A closer examination of the errors revealed that many of the words not found in the lexicon were erroneously tagged. The system could neither predict the correct tags for unknown words using the set of lexical rules nor did the contextual rules change the tags to correct ones” [28].

A logical response to such a problem was to improve the lexicon to contain the unknown words, which helped the tagger perform better. “It seemed reasonable to try to improve the performance of the tagger by extending the lexicon so that fewer word-forms in a new text would be unknown to the system” [28]. Prütz extended his lexicon to contain words that were previously unknown to the system, and it began to improve its performance. Prütz [28] said, “it is, however, possible to improve the result by extending the lexicon used by the tagger and thereby limiting the number of unknown words it has to deal with.”

Had Brill’s tagger been able to create a lexicon from the unannotated text provided, then Prütz would have spent much less time working with the tagger and trying to fix it. The unknown words that Prütz [28] identified as causing errors would not have been an exception, and his effort would have been substantially reduced. Assuming that it is possible to generate a quality lexicon solely from unannotated text, the accuracy results would be better than having a limited lexicon and unknown words.

B. Proposed Algorithm Introduction

The proposed algorithm is intended to generate a lexicon that can then be used by any POS tagger. The goal is to create this lexicon based on only an unannotated corpus. With each unannotated corpus presented to the generator a new lexicon will be created. By doing this, the lexicon will never contain any unknown words because it is generated from the same text the system is trying to tag. The idea is that if all words are considered unknown from the beginning, characteristics in the corpus will allow them to be grouped into categories of similar usage. Given these categories, a lexicon can be generated.

Devising such an algorithm requires many components. These components include: a standard set of data, a way to evaluate performance, and a way to generate the lexicon itself. A lexicon generator on its own is of little value; it is only useful when used in combination with a POS tagger. Therefore, a POS tagger and an existing lexicon have been selected to evaluate, i.e. use as a baseline for, the generated lexicon.

C. Components

The method for evaluating the system includes the following components: a hand-created lexicon, a generated lexicon, a corpus, and a part-of-speech tagger.

1. The part-of-speech tagger is a program that takes as input, a corpus and a lexicon. The tagger produces as output, a tagged version of the corpus containing what it thinks is the correct part-of-speech for each word. For this research, the unsupervised Brill tagger [10] has been selected. The output of the tagger is the measured dependent variable for the secondary hypothesis, and improvements in this area have been used as the measurement for success.

2. The corpus is the set of documents provided to the system that are in need of part-of-speech tagging. Ideally, this corpus can be any set of documents. However, for this research the Brown corpus from the Penn-TreeBank was used.

3. The hand-created lexicon acts as a dictionary, in theory it should contain all the words in the English language, however, for this research the Penn-TreeBank lexicon was used. This lexicon is the dependent variable for the primary hypothesis.

4. The generated lexicon is the independent variable in the experiment for the secondary hypothesis. This lexicon is created from the corpus provided. Therefore, the generated lexicon is a subset of the hand-created lexicon.

1) Part-of-Speech Tagger

The part-of-speech tagger chosen for this research is Eric Brill’s unsupervised tagger [10]. Eric Brill is a well-known computer scientist in the field of Natural Language Processing (NLP) because of his work developing both a supervised and unsupervised part-of-speech tagger. His unsupervised tagger has been evaluated as the best part-of-speech tagger, and it is widely, and freely, available. Other taggers, which are less well known, are also available; however, all of these taggers also require a lexicon in order to perform part-of-speech tagging. Since these freely available part-of-speech taggers require a lexicon, the intent is to be able to generate such a lexicon to bypass the dependence on a pre-existing one.

2) Corpus

In this study, to measure the success of this new lexicon generation procedure, the same corpora used to evaluate Brill’s tagger were obtained – the Wall Street Journal corpus and the Brown corpus from the Penn-TreeBank. The Brown
corpus has been selected for evaluating this system. The significance of using the Brown corpus lies in its wide availability and its familiarity to researchers in NLP. Because it was created as a representative sample of the English language, it is considered unbiased. Additionally mitigating any bias, selecting the Brown corpus from the Penn-TreeBank allows others to reproduce the results presented here, including a three-way numeric accuracy comparison performed on:

- the modified system (the experimentation)
- the Brill system (the benchmark)
- and the manually tagged corpus (the control case)

The Penn-TreeBank can be obtained through the LDC, Linguistics Data Consortium [19], the organization that controls its distribution. The LDC charges for the use of their corpora, and the Penn-TreeBank license costs $2,500. However, for academic research the license fee was reduced to $250.

3) Hand-Created Lexicon

The selected POS tagger, the unsupervised Brill tagger [10], uses the corpora available in the Penn-TreeBank as its annotated corpora for deriving its transformation-based rules. In addition, the tagger is dependent on the Penn-TreeBank in order to produce the “hand-created lexicon” mentioned earlier. This “hand-created lexicon” was produced by taking the words in the annotated Penn-TreeBank corpora and retaining a list for all the uses of each word. An example is shown in Section III. F. 1 – “Establishing the Control Case.”

D. Improvements

The improvements proposed for this research involve making the generated lexicon more accurate, but this is not a mutually exclusive improvement. The more important improvement is the accuracy of the POS tagger using the automatically generated lexicon. Since the POS tagger is dependent on the lexicon, the only way to improve the accuracy of the tagger is to improve the accuracy of the lexicon.

E. Process

Thus, using the Penn-TreeBank as the data source and the unsupervised Brill tagger as the part-of-speech tagger, the research process is defined as follows:

- The control case is the fixed set of manually annotated data provided by the Penn-TreeBank.

The following are evaluated based on the control case above:

- The benchmark is the resulting accuracy of the Brill tagger given the Penn-TreeBank “hand-created lexicon.” Brill [10] claims 97% accuracy with his tagger on this data set.
- The experimentation measurement is the resulting accuracy of the Brill tagger given the automatically generated lexicon.

Using a more formal notation to represent the hypothesis and each of the dependent and independent variables, H is used to represent each of the hypotheses 1 and 2, IV is used to represent the independent variable for each of the hypotheses, and DV is used to represent the dependent variable for each of the hypotheses.

Description of hypothesis 1 and its variables:
H1: Generated lexicon, using no external information, is comparable to the general-purpose Penn-TreeBank Brown corpus lexicon
IV1: Assignment of POS to each term in the generated lexicon
DV1: Assignment of POS to each term in the hand-created lexicon

H1 defined by its variables:
DV1 ≈ (IV1 = f(most frequent character, context, morphology, grouping terms, assigning terms))

Description of hypothesis 2 and its variables:
H2: Accuracy of POS tagger output is similar when given same corpus but different lexicons
IV2: Lexicon (general-purpose or generated)
DV2: Accuracy of POS tagger

H2 defined by its variables:
Given: DV2i = f(POS_Tagger(IV2i))
DV2(general-purpose lexicon) ≈ DV2(generated lexicon)
There were several important decisions that needed to be made during the course of this research. Many of those decisions were made before any of work could begin, for example, deciding to use and obtain the Penn-TreeBank was an important decision. Another important decision made at the outset of this research was the selected tag-set. Since the attempt to create an automatic lexicon generator is a novel idea, the problem was constrained by reducing the size of the tag-set. As described in the section above, about the Penn-TreeBank, there are an infinite number of tag-sets, of which the Penn-TreeBank has selected one. Yet, a problem arises by using the Penn-TreeBank tag-set – there are 36 parts-of-speech tags, each having a special form and a corresponding special meaning.

The approach taken in this research was not to use any special tags with corresponding special meanings. Instead, the approach was to use a categorization scheme based on numeric values from 1 to 7. Choosing such a categorization scheme allows for the addition of more tags in the future by simply adding to the list sequentially. Seven categories were chosen because the Penn-TreeBank tag-set can be easily reduced into the following categories: category 1: conjunctions, determiners, and other short words, category 2: numbers or symbols, category 3: adjectives, category 4: nouns, category 5: pronouns, category 6: adverbs, and category 7: verbs. By constraining the problem, the intent was to conceptualize the algorithm in an easier way.

1) Establishing the Control Case

In order to generate the control case and evaluate the results of the lexicon generator with the “correct” answer, transformations were performed on the Penn-TreeBank lexicon. The transformations occurred by creating an equivalence class for each part-of-speech tag. Then, for each word, its list of tags was transformed into those equivalence classes. The new tags were then reduced to remove redundancy and sorted for ease of readability. This provided the transformed version of the Penn-TreeBank lexicon. An example using the word “the” shows this transformation:

The entry in the Penn-TreeBank lexicon
the DT VBD VBP NN|DT IN JJ NN NNP PDT
Perform the transformation
the 1 7 4|1 1 3 4 4 3
Reduce and sort the transformation
the 1 3 4 7

Below is the list of 36 POS tags defined in the Penn-TreeBank and how they were assigned to the seven categories for this approach.
<table>
<thead>
<tr>
<th>Penn-TreeBank</th>
<th>Dennis Pereira’s grouped</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TAGS WITH CORRESPONDING PART OF SPEECH [29]</td>
<td>LIST OF TAGS WITH CORRESPONDING PART OF SPEECH</td>
</tr>
<tr>
<td>1. CC Coordinating conjunction</td>
<td>Category 1 – Conjunctions, determiners, short words</td>
</tr>
<tr>
<td>2. CD Cardinal number</td>
<td>1. CC Coordinating conjunction</td>
</tr>
<tr>
<td>3. DT Determiner</td>
<td>3. DT Determiner</td>
</tr>
<tr>
<td>4. EX Existential there</td>
<td>4. EX Existential there</td>
</tr>
<tr>
<td>5. FW Foreign word</td>
<td>5. FW Foreign word</td>
</tr>
<tr>
<td>6. IN Preposition or subordinating conjunction</td>
<td>6. IN Preposition or subordinating conjunction</td>
</tr>
<tr>
<td>7. JJ Adjective</td>
<td>16. PDT Predeterminer</td>
</tr>
<tr>
<td>8. JJR Adjective, comparative</td>
<td>23. RP Particle</td>
</tr>
<tr>
<td>9. JJS Adjective, superlative</td>
<td>25. TO to</td>
</tr>
<tr>
<td>10. LS List item marker</td>
<td>26. UH Interjection</td>
</tr>
<tr>
<td>11. MD Modal</td>
<td>33. WDT Wh-determiner</td>
</tr>
<tr>
<td>12. NN Noun, singular or mass</td>
<td>Category 2 – Numbers and symbols</td>
</tr>
<tr>
<td>13. NNS Noun, plural</td>
<td>2. CD Cardinal number</td>
</tr>
<tr>
<td>14. NNP Proper noun, singular</td>
<td>5. FW Foreign word</td>
</tr>
<tr>
<td>15. NNPS Proper noun, plural</td>
<td>10. LS List item marker</td>
</tr>
<tr>
<td>16. PDT Predeterminer</td>
<td>24. SYM Symbol</td>
</tr>
<tr>
<td>17. POS Possessive ending</td>
<td>Category 3 – Adjectives</td>
</tr>
<tr>
<td>18. PRP Personal pronoun</td>
<td>7. JJ Adjective</td>
</tr>
<tr>
<td>19. PRPS Possessive pronoun</td>
<td>8. JJR Adjective, comparative</td>
</tr>
<tr>
<td>20. RB Adverb</td>
<td>9. JJS Adjective, superlative</td>
</tr>
<tr>
<td>21. RBR Adverb, comparative</td>
<td>Category 4 – Nouns</td>
</tr>
<tr>
<td>22. RBS Adverb, superlative</td>
<td>12. NN Noun, singular or mass</td>
</tr>
<tr>
<td>23. RP Particle</td>
<td>13. NNS Noun, plural</td>
</tr>
<tr>
<td>24. SYM Symbol</td>
<td>14. NNP Proper noun, singular</td>
</tr>
<tr>
<td>25. TO to</td>
<td>15. NNPS Proper noun, plural</td>
</tr>
<tr>
<td>26. UH Interjection</td>
<td>17. POS Possessive ending</td>
</tr>
<tr>
<td>27. VB Verb, base form</td>
<td>Category 5 – Pronouns</td>
</tr>
<tr>
<td>28. VBD Verb, past tense</td>
<td>18. PRP Personal pronoun</td>
</tr>
<tr>
<td>29. VBG Verb, gerund or present participle</td>
<td>19. PRPS Possessive pronoun</td>
</tr>
<tr>
<td>30. VBN Verb, past participle</td>
<td>34. WP Wh-pronoun</td>
</tr>
<tr>
<td>31. VBP Verb, non-3rd person singular present</td>
<td>35. WPS Possessive wh-pronoun</td>
</tr>
<tr>
<td>32. VBZ Verb, 3rd person singular present</td>
<td>Category 6 – Adverbs</td>
</tr>
<tr>
<td>33. WDT Wh-determiner</td>
<td>11. MD Modal</td>
</tr>
<tr>
<td>34. WP Wh-pronoun</td>
<td>20. RB Adverb</td>
</tr>
<tr>
<td>35. WPS Possessive wh-pronoun</td>
<td>21. RBR Adverb, comparative</td>
</tr>
<tr>
<td>36. WRB Wh-adverb</td>
<td>22. RBS Adverb, superlative</td>
</tr>
<tr>
<td></td>
<td>36. WRB Wh-adverb</td>
</tr>
<tr>
<td></td>
<td>Category 7 – Verbs</td>
</tr>
<tr>
<td></td>
<td>27. VB Verb, base form</td>
</tr>
<tr>
<td></td>
<td>28. VBD Verb, past tense</td>
</tr>
<tr>
<td></td>
<td>29. VBG Verb, gerund or present participle</td>
</tr>
<tr>
<td></td>
<td>30. VBN Verb, past participle</td>
</tr>
<tr>
<td></td>
<td>31. VBP Verb, non-3rd person singular present</td>
</tr>
<tr>
<td></td>
<td>32. VBZ Verb, 3rd person singular present</td>
</tr>
</tbody>
</table>
Possible Problems

Performing the transformation from one tag-set to another may have a negative influence on the results. Since Brill’s tagger uses contextual rules to identify which tag to use in a given case, if the set of tags is reduced it could also be argued that the rule accuracy will be reduced. This case has been addressed, and is described in the experimentation section later. The results of the Brill tagger are degraded due to the transformations, but the results remain within a reasonable bound.

Proof-of-Concept (previous work)

The process of developing the algorithm for the lexicon generator began early-on with a very simple approach. A proof-of-concept automatic lexicon generator was demonstrated by Pereira in 2003 [26]. The proof-of-concept outlines an algorithm for generating a lexicon and shows the ability to perform a simple grouping of words. However, the performance of the proof-of-concept was poor and needed further research to be improved.

To measure the usefulness of the proof-of-concept, the four components mentioned above were used to obtain a score for two POS tagger runs. While the first run used the existing “hand-created lexicon”, the second run used the “generated lexicon.” The difference between the proof-of-concept and the current system can be found in the corpus selected. The proof-of-concept used some randomly selected news articles from Yahoo! News. Because the corpus was originally unannotated, there was no control case. Therefore, a control case was created using the first run of the POS tagger. When the tagger was run on the corpus using the “hand-created lexicon” it became the baseline. An accuracy score was obtained by comparing the result of second run to the result of the first. Using the generated lexicon resulted in a 60.8% accuracy.

Proof-of-Concept Algorithm (previous work)

As mentioned before, the algorithm proposed in the proof-of-concept was a simple one. It involved taking the corpus of untagged text and performing some simple frequency statistics on it. The most frequent character in the unannotated text is identified, which, in English, happens to be the space character. The corpus is then split-up based on that most frequent character, producing what this paper defines as “terms.” For English, these terms happen to be words, but for other languages this may not be the case because the space may not be the most frequent character. Once the terms have been identified, they are collected and counted to find the most frequent one. For English, the most frequent term is “the.”

The next step is the most critical; it is in this step that the categorization process begins. The seven most frequent terms are assigned, by default, to category one – the category containing conjunctions, determiners, and other short, and very frequent words. The English corpus in the proof-of-concept (i.e. Yahoo! News) contained these seven most frequent words: “the”, “of”, “to”, “and”, “a”, “in”, “that”, and “said”. The algorithm continues the categorization process by taking those most frequent terms and finding the terms that are immediately to the left and to the right of those terms. Those terms are assigned to two categories – four and seven. These two categories happen to be the noun category and verb category. Lastly, the least frequent terms are identified, specifically those terms that appear only once (hapax words as defined by Mikheev [22]), and are assigned to category four (nouns).

Brill [10] stated that, “…nouns are much more common than verbs or modals,” and thus nouns are likely to be mentioned only once, while other forms of speech are likely to be repeated in a large corpus. It is for this reason that hapax terms are assigned to category four. However, as the corpus size increases, the assignment of hapax terms will need to be expanded due to the higher likelihood of an obscure word appearing more than once. This problem can be addressed in the future by using the logarithm of the number of terms in the corpus to determine which ones qualify as hapax-like terms.

The justification for categorizing, into categories four and seven, the terms immediately to the left and to the right of the most frequent terms comes from the assumption that most words in English are either a noun or a verb. As Brill [10] said, it is more likely for nouns to appear than verbs, but if a word is not a noun, Pereira [26] makes the assumption that it will likely be a verb. The results were promising enough to merit this further work.

Lexicon Generation Algorithm

Since the proof-of-concept algorithm was so simple, it was not capable of performing an acceptable job in assigning terms to their proper categories, nor was it expandable to improve certain portions of the algorithm that would cause it to perform better. The proof-of-concept algorithm has since been improved, resulting in the “selected algorithm,” which is much more complex than the one presented above. The selected algorithm provides a framework for establishing future research and allows for individual portions of the algorithm to be modified so results can be improved. The selected algorithm incorporates concepts such as context and morphology as proposed by Orphanos [25], Thede [33], Nakagawa [23], and Mikheev [22]. Additionally, concepts taken from Cucerzan [13] provide a basis for using only the unannotated text to generate the lexicon. The primary difference between any of the approaches mentioned above is the absence of pre-existing data (i.e. a lexicon or training data).

Selected Algorithm

The selected algorithm is based on the proof-of-concept and therefore incorporates some of the same methods for achieving the final goal, lexicon generation. The first step remains the same, identifying the most frequent character. However, to provide a better frame of reference, the exact
procedure will now be described. A corpus is contained in a single text file, which is fed into the algorithm. The algorithm takes the corpus as input and retains a sorted list of each character and its counts. The most frequent character is the first item in the list, and once the entire corpus has been scanned, this character will play a significant role in the next step of the algorithm.

The second step is dividing the corpus into “terms,” which can also be called tokenization. Because of its limitless idiosyncrasies, tokenization is a field that could be studied on its own, as shown by Branco and Silva [6]. The tokenization approach used in this paper is a simple one, capable of handling languages with words that are distinguishable by surrounding spaces. In English, the most frequent character in a corpus is the space, and so it is used here to separate one “term” from another.

2) Morphology

The third step, which in the proof-of-concept was the most critical, is now different. Instead of counting the terms and selecting the seven most frequent ones, a new method is proposed to identify frequent morphology. Brants [7], Thede [33], Curczerzan [13], and Mikheev [22] all used some variation of prefix and suffix analysis, in which some look for a root or stem word that can have various other affixes attached to it. Unlike their approaches, this algorithm is based on the assumption that any sequence of characters is useful. The TNT POS tagger [21] uses up to ten suffix characters to analyze unknown words while other POS taggers use only four suffix characters. In order to provide a more general solution, all the characters in the corpus are considered. Every combination of characters contained in the corpus greater than or equal to length two and less than or equal to length seven are considered.

This step operates independently from step two; thus, they are interchangeable. Analysis of all the characters is performed by creating “n-graphs,” which are substrings of the given corpus. The term “n-graph” is a mutation of the term “n-gram” [34], which describes an algorithm based on words. The “n-graph” concept in this algorithm applies to sequences of characters of length two through seven, which can include words or parts of words. The algorithm collects every sequence of characters of length two, three, four, five, six, and seven, retaining the counts (i.e. frequency) for each sequence. This counting is identical to the counting performed in step one for the most frequent character, except that the items counted are now sequences, rather than individual characters. Once the entire corpus has been scanned, the 20 most frequently appearing character sequences of each length are retained. It is important to note that the character sequences do not include the most frequent character (MFC); therefore, in this algorithm none of the character sequences contain the space character.

At first, all character sequences were considered equivalent, but this did not provide the desired results. To address this problem, priority was given to character sequences found in prefix and suffix positions. Specifically, character sequences in prefix and suffix positions are counted separately from the same character sequence found elsewhere in the corpus. For example, the sequence “ed”, found in the words “booked” and “looked”, is counted separate from the sequence “ed” found in the words “ledge” or “hedge.” Using this approach it is possible to find the common prefixes and suffixes used within a language – a substantial basis for grouping terms.

The result of this part of the algorithm produced 120 “n-graphs” which were used to create collections of terms. The 120 “n-graphs” are made up of the 20 most frequent “n-graphs” for each of the six character sequence lengths (two
The most frequent 120 “n-graphs” produced from the Brown corpus are shown in Table 4. The “n-graphs” were used to identify words containing similar morphology.

The fourth step is also part of this morphology section. It involves finding all the terms in the corpus that contain each “n-graph” so a collection can be created. The algorithm starts with the longest and most frequent “n-graph” and looks for terms containing it. When a term containing the “n-graph” is found, it is assigned to a collection, and once assigned it cannot be re-used. The algorithm iterates sequentially through the “n-graphs” until the first set of 20 “collection-generating n-graphs” are reached. For a “collection-generating n-graph,” there is a condition placed on the sequential process. If the current “n-graph” does not produce any members, then the following “n-graph” is considered. This situation can be exemplified by the sequence “hich” which does not produce any members because all of its potential members were assigned by a sequence appearing before it – “which.” Once all 20 “n-graphs” of length seven have been considered, the process is repeated for n-graphs of length six, five, four, three, and two. The resultant output is saved in a file and used later in the algorithm.

3) Context

The fifth step in the algorithm builds on the steps described for morphology. Since 120 collections were created using morphology, a method for collapsing them is needed. Many of the collections are of the same category, and the ability to group them allows us to generate a lexicon. Using the context found in the corpus, a solution is provided. The need to combine the groupings arose because there were too many collections and they needed to be placed into 7 categories. The most effective solution was to use context to group words used in similar ways. Such words are likely to retain the same part of speech just as in the “yarzbygu” and “giraffe” examples given earlier. Mikheev [22] did not use context in his strategy; instead, he tried to improve the lexicon based on an existing lexicon, rather than on a corpus. Unlike Mikheev [22], Brill [10] and Cucerzan [13] used context to determine how to categorize unknown words. This supports the basis of using context as a means of grouping. Other clustering techniques described earlier were considered, but context was found to be the most useful.

The algorithm for gathering the context from the corpus involves collecting a context of three terms – based on work by Brill [10] and Brants [7]. A window slides over the text to collect every set of three sequential terms, then the term immediately to the left and to the right are added to its context list. If the same set of terms is encountered again after they have already been added to the context list, they will not be re-added. When this occurs, only the new terms appearing next to the context will be added. A count is retained for each term appearing in the same context. Once the entire corpus is scanned a listing is created containing each three-term context and the terms that appear to the left and to the right of that context. Once this list is created it is saved in a file for later use.

4) Combining Context and Morphology

The sixth step involves combining the context list with the morphology list to produce a resultant set of groupings. The goal is to reduce the number of collections in an attempt to end up with 7 groupings that would represent the 7 categories. Unfortunately, this goal was not accomplished; however, several useful groupings were identified. The algorithm for joining the context with the morphology is as follows: the morphology is loaded into memory and a table is created listing each term and its “collection-generating n-graph”, then the context is processed beginning with the words appearing with the first context. Terms appearing to the left and right of the context are considered separately. The first set of terms appearing in the same context are all assigned to group 1. Additionally, all the words that contain the same “collection-generating n-graph” are also assigned to group 1. When the second set of terms is considered, its contents will be assigned to group 2. If a term has already been assigned it cannot be reassigned, therefore, the assignments to groups are quickly exhausted. In order to quickly reduce the number of collections, a heuristic was imposed that only considered four groupings, this entailed assigning three categories and then assigning all other terms to the fourth. This minimized the challenge of assigning terms to the 7 selected POS tag categories because two of them could be defined as closed sets – category 1 (conjunctions, determiners, etc.) and category 2 (numbers or symbols). This would leave only 5 categories that need to be generated. The four generated groupings did not correspond accurately to the 5 remaining categories, but they were evenly distributed so that the groupings could be assigned a different way, producing some interesting results.

The seventh and last step is to empirically assign the categories generated by the combination of context and morphology to categories. Based on the resulting 4 groupings, the results were analyzed and the groups were assigned to categories. The result of this assignment was groups 1 and 2 being assigned to category 7 and groups 3 and 4 being assigned to category 4. The terms that had not been assigned by either context or morphology were assigned to category 4, the 7 most frequent terms were assigned to category 1, and terms containing numbers were assigned to category 2.

It is important to note that improvements can be made to the assignment process to improve the results. To see the results of the combining step without any heuristic imposed see Appendix A. Contained in the appendix is the full list of term groupings along with their correct answer (based on the Penn-TreeBank lexicon). A total of 98 categories were generated, and many of them can be clearly identified as sets...
of existing categories. An additional grouping stage may be needed between steps six and seven in order to take advantage of these more robust groupings.

5) Summary
To summarize the algorithm, a graphic has been created and presented in Figure 10. The input is the corpus and the output is the lexicon. The algorithm is contained in the 7 steps in between.

I. Alternative Algorithms
In leading up to the selected algorithm several paths were taken before the algorithm presented above was selected. Five other algorithms were considered, some for comparison purposes only, and others were attempts to provide a solution to the lexicon generation problem but were not selected in the end.

1) Proximity Algorithm
One of the unselected algorithms tries to take advantage of the proximity of terms to other terms in combination with their morpho-syntactic characteristics such as length and frequency. The algorithm keeps track of each “term” (aka token, word) appearing in the text. It could be considered a database that retains each term’s length and frequency, along with its position in the corpus - similar to a text-retrieval index. Additionally, the 7 most frequent words are identified. By Zipf’s Law [18] there are only a few highly-occurring words and many less-occurring words. Traditionally, the most frequent words are often called “stop words”, and are disregarded from further processing because they are considered to have little functional value or grammatical meaning. However, in this approach the hypothesis was that these words might be useful in detecting other patterns that could allow for the categorization of terms. Harald Baayen [2] showed that “high-frequency words have more neighbors than low frequency words,” which may seem obvious, but more importantly he showed that “high-frequency words have higher-frequency neighbors than low-frequency words.” He called these the neighborhood density and neighborhood frequency effects [2].

Using the length, frequency, substring characteristics, and proximity to one of the 7 most common terms, a single calculation was performed to combine all the available data. The hope was to use some of the proximity (neighborhood) ideas [18] to perform some calculations that would be used as predictors. Essentially, the question to be answered was: could only the 7 most frequent terms be used as a predictor to categorize the terms? To answer this question, the following method was used. Given a term; for each time it appeared in the corpus, the nearest “frequent term” was found to the left and right of it. A list was created as shown below:

L6 R7 successfully
[the 8 to 7 of 5 and 5 that 3 in 3 ]
[the 12 in 7 and 5 to 4 of 3 ]

The list above contains the neighbors, their frequencies, and their combined average distance from the word “successfully.” In the Brown corpus, the most common neighbor “the” appears 8 times to the left of “successfully.”
and 12 times to the right. A calculation was performed over all the neighbors to find the average distance between any one of them and the word “successfully.” The result was an average distance of 6 terms on its left and 7 terms on its right. Using this neighbor information combined with the other morpho-syntactic characteristics, a calculation was performed. Unfortunately, no combination of the measurements used could offer an acceptable way of categorizing the terms. Additionally, this approach had very little room to grow in terms of improving accuracy by changing specific parts of the algorithm. Because of its unacceptable results and its limited expansion, this algorithm was not selected.

2) Length-Frequency Algorithm
Another algorithm that was considered, but not selected, is the “length-frequency algorithm.” This algorithm attempted to use the length and frequency of a term to assign it to the appropriate categories. The foundation for performing its categorization is based on empirical knowledge derived from the Brown corpus, shown below:

<table>
<thead>
<tr>
<th>Category</th>
<th>Average Length</th>
<th>Median Length</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat1</td>
<td>4.98 chars</td>
<td>5.00 chars</td>
<td>1-17</td>
</tr>
<tr>
<td>cat2</td>
<td>6.11 chars</td>
<td>6.00 chars</td>
<td>1-32</td>
</tr>
<tr>
<td>cat3</td>
<td>10.29 chars</td>
<td>10.00 chars</td>
<td>1-54</td>
</tr>
<tr>
<td>cat4</td>
<td>7.87 chars</td>
<td>7.00 chars</td>
<td>1-40</td>
</tr>
<tr>
<td>cat5</td>
<td>4.02 chars</td>
<td>4.00 chars</td>
<td>1-10</td>
</tr>
<tr>
<td>cat6</td>
<td>8.67 chars</td>
<td>9.00 chars</td>
<td>1-20</td>
</tr>
<tr>
<td>cat7</td>
<td>7.61 chars</td>
<td>7.00 chars</td>
<td>1-22</td>
</tr>
<tr>
<td>cat8</td>
<td>38.76</td>
<td>3.00</td>
<td>1-62474</td>
</tr>
</tbody>
</table>

The table above shows length and frequency metrics taken from the annotated Brown corpus. Average and median lengths and frequencies were calculated, and the range for each measure and category is also shown. Using this information rules were created to assign terms to categories. The rules are:

Assign to cat1 if frequency > 700 and length < 7
Assign to cat2 if frequency > 70 and frequency < 150 and length < 8
Assign to cat3 if frequency > 20 and frequency < 40 and length > 7
Assign to cat4 if frequency > 0 and frequency < 25 and length > 5
Assign to cat5 if frequency > 500 and frequency < 700 and length < 6
Assign to cat6 if frequency > 90 and frequency < 250 and length > 6
Assign to cat7 if frequency > 25 and frequency < 50 and length < 10

If the term cannot be assigned by one of the previous rules then assign it to category 4.

The result of this algorithm was very poor and justified its dismissal as a possible solution for lexicon generation. It also showed that length and frequency alone are not good predictors of a term’s category.

3) Probabilistic Algorithm
Another algorithm implemented and used as a baseline was the “probabilistic algorithm.” This algorithm uses the probabilities presented below to randomly assign terms to a certain category until its probability has been reached. However, since a term is capable of playing multiple roles, the category probabilities sum up to more than the size of the entire lexicon. This anomaly is regarded as insignificant and was resolved by collecting terms into categories until their calculated probability has been reached, or until all the terms have been assigned. The probabilities are shown below as percentages of the lexicon size.

<table>
<thead>
<tr>
<th>Category</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat 1</td>
<td>0.79</td>
</tr>
<tr>
<td>Cat 2</td>
<td>4.26</td>
</tr>
<tr>
<td>Cat 3</td>
<td>18.82</td>
</tr>
<tr>
<td>Cat 4</td>
<td>60.20</td>
</tr>
<tr>
<td>Cat 5</td>
<td>0.20</td>
</tr>
<tr>
<td>Cat 6</td>
<td>4.12</td>
</tr>
<tr>
<td>Cat 7</td>
<td>19.33</td>
</tr>
<tr>
<td>Sum</td>
<td>107.72</td>
</tr>
</tbody>
</table>
This algorithm produced poor results and indicated that mere probabilities of the categories are not sufficient predictors of a term’s category.

4) Random Algorithm
A fourth algorithm implemented and used as a baseline was the “random algorithm.” This algorithm came in two flavors, single assignment and multiple assignment. The first version of this algorithm randomly selects a single category (1-7) for each term. The result is the generated lexicon with terms assigned to exactly one category. The second version of this algorithm randomly assigns terms to multiple categories. This is accomplished by randomly selecting a number between 1 and 6, which corresponds to the number of categories that will be assigned to the term. The reason for selecting 6 instead of 7 for the possible number of assigned categories is because the Brown corpus does not contain any terms that are assigned to all 7 categories. Only five terms fall into 6 categories, even though four of them fall into punctuation categories. The terms that fall into 6 categories are:

'.' 1 2 3 4 7 ;
con 1 2 3 4 6 7
that 1 3 4 5 6 7
up 1 3 4 6 7
]* 1 3 4 6 7

Once the number of categories has been identified (1-6) for a term, then it is randomly assigned to that number of categories. For example, if a term is randomly assigned to 3 categories, then those 3 categories are also randomly assigned. The two versions of this algorithm were used as a comparison point for the selected algorithm. The results for this algorithm were so poor that an additional comparator of better accuracy needed to be created.

5) Noun Algorithm
The final algorithm used for comparison against the selected algorithm turns out to be the most simple to describe. This algorithm, the “noun only algorithm,” performs better than either of the two other comparison algorithms (probabilistic and random). Its approach is to assign all the terms to a single category, category 4 (nouns).

IV. EXPERIMENTATION

A. Overview
When evaluating the results of the algorithm proposed in this paper, there are two measurements that need to be considered. The first measurement is the correctness, or accuracy, of the lexicon that is being generated, while the second is the accuracy of the POS tagged output. The accuracy of the lexicon is calculated by comparing the generated lexicon with the Penn-TreeBank lexicon. The accuracy of the tagged output is calculated by comparing the output of the Brill tagger with the manually annotated version of the Penn-TreeBank Brown Corpus. In short, the two distinct comparisons are on the lexicons and on the annotated corpora.

B. Evaluating Lexicon Accuracy
Some of the algorithms presented in the previous section as alternatives to the lexicon generation algorithm, are useful for baseline comparisons with the proposed solution. Four algorithms have been selected as baseline comparators, they are: the “probabilistic algorithm,” the two versions of the “random algorithm,” and the “noun-only” algorithm. All the experiments for the baseline lexicon algorithms were conducted in the same manner. Each one was considered a substitute for the independent variable for hypothesis 1 (IV1).

The lexicons were created by an implementation of each algorithm. Then, each lexicon was measured for accuracy against the Penn-TreeBank lexicon (DV1).

The accuracy for IV1 is measured in two ways, the first is an exact match score that compares the correct answer from the Penn-TreeBank lexicon to the generated answer; if they match then the correct count and the total count is incremented, otherwise only the total count is incremented. The second way of measuring accuracy compliments the first but is more relaxed, allowing for “partial matches.” Because the lexicon consists of a set of tags for each word, this second approach checks to see if only part of the correct answer was generated. To provide an example, the lexicon entry for the word “win” is “win 7 4”. In this case, if the lexicon generator produced an entry “win 7” then it would be counted as correct, however, under the first method of scoring it would be incorrect. Having these two methods for scoring results in two accuracy measures, one consisting of only exact matches and the other consisting of a summation of exact matches and partial matches.

1) Baseline Algorithm Accuracies
The first baseline comparator is the “probabilistic algorithm.” This algorithm randomly distributes its category assignments based on the Penn-TreeBank’s correct assignment. An exact description of this algorithm can be found in the previous section. The results of this algorithm are in the middle of the pack for the baselines, with an exact match accuracy of 23.7% and a combined, exact and partial match, accuracy of 50.2%.

The “random algorithm – terms assigned to a single category” is the second comparator. In this algorithm, each term was randomly assigned to exactly one category. When compared to the correct lexicon, the result was an exact match accuracy of 11.9% and a combined accuracy of 15.9%.

The third comparator is the “random algorithm – terms assigned to multiple categories.” In this algorithm, each term is randomly selected to have N < 7 categories, then those N distinct categories are randomly selected and assigned to the term. As reported earlier, this approach resulted in a much lower accuracy, 2.2% exact match and 2.9% combined match. The reason for such a low score is analogous to a lottery situation where the number of winning numbers ranges between 1 and 6, and the available numbers range from 1 to
7. In other words, instead of picking 5 numbers, the case would be, pick between 1 and 6 numbers, with the numbers available for selection ranging between 1 and 7. Since the number of categories is unknown, the probability of randomly selecting the appropriate number of correct categories significantly reduces the accuracy of the lexicon.

The fourth, and last, comparator is the “noun only algorithm.” This algorithm, as described earlier, simply assigns all terms to category 4. The accuracy measures for this algorithm are the highest of all the baseline comparisons, with an exact match accuracy of 53.0% and a combined match accuracy of 65.0%.

2) Lexicon Generation Accuracy

The purpose for selecting these four baseline comparisons is to show the following: that the lexicon generation algorithm performs better than a completely random distribution; it performs better than a distribution based on the category probabilities found in the Brown Corpus; and, it performs better than simply assigning everything to a single category. The lexicon generation algorithm was evaluated (H1) in the same manner as the baselines described above, and the resulting accuracies were 56.1% exact match and 70.0% combined match.

3) Discriminant Analysis

To validate the results of the lexicon generation algorithm (H1) from another perspective, a discriminant function analysis was used. The generated category “score” based on four groupings was used as the predictor for the original seven categories on a subset of 32,784 terms selected because they were assigned to a single POS category by the Penn-TreeBank lexicon. Overall, the statistical functions created via this process correctly classified 58.2% of the terms. Nouns (category 4), which comprised over half of the terms (55.5%), were correctly classified in 96% of the cases in both the first analysis and the cross-validated case (see Table 5). Almost one third of the pronoun set (category 5) was correctly classified (30.8%), but only 23.3% of the verbs (category 7) were correctly classified. The territorial map shows how the first two discriminant functions separate or "discriminate" terms in categories 4 (nouns), 5 (pronouns), and 7 (verbs). Below are both the results table and the territorial map of the first two discriminant functions.

<table>
<thead>
<tr>
<th>Classification Results</th>
<th>Predicted Group Membership</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>1.00</td>
</tr>
<tr>
<td>Original Count</td>
<td>1.00</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>6.00</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>7.00</td>
<td>24</td>
</tr>
<tr>
<td>%</td>
<td>1.00</td>
<td>17.6</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>20.5</td>
</tr>
<tr>
<td></td>
<td>6.00</td>
<td>.9</td>
</tr>
<tr>
<td></td>
<td>7.00</td>
<td>.4</td>
</tr>
<tr>
<td>Cross-validated Count</td>
<td>1.00</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>6.00</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>7.00</td>
<td>24</td>
</tr>
<tr>
<td>%</td>
<td>1.00</td>
<td>17.6</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>3.00</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4.00</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5.00</td>
<td>20.5</td>
</tr>
<tr>
<td></td>
<td>6.00</td>
<td>.9</td>
</tr>
<tr>
<td></td>
<td>7.00</td>
<td>.4</td>
</tr>
</tbody>
</table>

a. Cross validation is done only for those cases in the analysis. In cross validation, each case is classified by the functions derived from all cases other than that case.

b. 58.2% of original grouped cases correctly classified.

c. 58.2% of cross-validated grouped cases correctly classified.

Table 5: Discriminant Analysis Classification Results
Symbols used in territorial map

Symbol  Group  Label
-----  -----  -------------------
1        1
2        2
3        3
4        4
5        5
6        6
7        7

* Indicates a group centroid
4) Comparing Lexicon Baselines to Proposed Algorithm

Even though the exact and combined accuracies of the lexicon generation algorithm are only slightly higher than the “noun only algorithm,” the next section will describe how the lexicon will be used with the POS tagger, and will compare the results of the POS tagger when it is run with different lexicons. Additionally, the proposed algorithm performs substantially better than the proof-of-concept in Pereira [26]. Pereira reported only the combined accuracy, 42%, showing that the proposed algorithm in this study has improved accuracy by nearly 30%. The table below shows the accuracy scores for the baseline lexicon generation algorithms compared with the proposed algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Correct</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilistic</td>
<td>23.7%</td>
<td>50.2%</td>
</tr>
<tr>
<td>Random Single Category</td>
<td>11.9%</td>
<td>15.9%</td>
</tr>
<tr>
<td>Random Multiple Categories</td>
<td>2.2%</td>
<td>2.9%</td>
</tr>
<tr>
<td>Noun Only</td>
<td>53.0%</td>
<td>65.0%</td>
</tr>
<tr>
<td>Lexicon Generator</td>
<td>56.1%</td>
<td>70.0%</td>
</tr>
</tbody>
</table>

Table 6: Baseline Algorithms vs. Lexicon Generator

C. Corpus Characteristics

The corpus used in this study was the Penn-TreeBank Brown Corpus. Comprised of 1,170,817 terms, the corpus is made up of 53,849 unique terms. These terms and their tags are represented in the lexicon and used by the unsupervised Brill tagger to produce part-of-speech tagged versions of a corpus. It is important to remember the systems were scored using a measure of accuracy, as presented earlier. The accuracy is based on the Penn-TreeBank’s manually tagged Brown Corpus. This corpus was painstakingly created, manually, to provide a baseline measurement in these types of experiments. Unfortunately, a few anomalies were found in the data; they are presented now along with the corrections made. The first anomaly is a single instance of the word “best” being tagged as JJSS, which is most likely a typographical error. This tag was changed from JJSS to JJS since it was the only occurrence of the tag in the entire corpus. The second anomaly was an undocumented tag that appeared in the corpus, namely the PRPSR tag, which was used to tag the word “her”. Since this tag was used consistently to tag every instance of the word “her”, it was therefore added to the list of possible tags and into category 5, i.e. pronouns.

1) Effect on Results by Modifying the Tagset

Once the baseline corpus was corrected, it was converted to represent the numerical categories, i.e. 1-7, used in this study. A concern that needed to be addressed was the impact of changing the tagset on the accuracy of the tagger. Given the Brown Corpus and its matching lexicon, running them through the Brill tagger resulted in an accuracy score of 94.2%. The converted corpus with its matching converted lexicon when run through the Brill tagger resulted in an accuracy score of 91.2%. The impact of changing the tagset is only 3%, which is considered acceptable for this experimentation.

Another important change that needs to be mentioned is the structure and make-up of the lexicon. In the Penn-TreeBank lexicon, after each term there is a list of category tags ordered first by the most likely tag followed by each succeeding less likely tag. Yet, the conversion process used in this study has been written to re-order the categories in the lexicon, therefore the advantage of knowing the most likely tag for each term is lost. The decision to disregard the original lexicon’s priority order was made because the proposed lexicon generation algorithm has no mechanism for assigning category priority. Still, by making this change, the accuracy score for the converted corpus and its matching ordered lexicon when run through the Brill tagger was 89.3%. The impact of sorting the terms results in a reduction of ~2%, which is also acceptable for this experimentation.

The lexicon used to perform the baseline experiments was produced from the correctly tagged Penn-TreeBank Brown Corpus only. This is to distinguish it from the existing lexicon that is packaged with the Brill tagger, which is a lexicon produced from both the Brown and Wall Street Journal corpora. The intent in packaging a lexicon with the tagger is to provide the user a starting point. Some additional baseline experiments were performed using this lexicon, and the results show that a larger lexicon has negligible impact on the accuracy of the tagger. However, when the lexicon is converted and the tags are re-ordered, there is a significant impact on the accuracy. When the Brown Corpus is run through the Brill tagger with the large converted lexicon the accuracy score was 76.1%, a reduction of more than 13% over the smaller lexicon. This shows that as a lexicon becomes more general, terms begin to have higher occurrences of more than one type. The table below shows the results of these experiments.

<table>
<thead>
<tr>
<th></th>
<th>Unconverted Lexicon</th>
<th>Converted Lexicon - Brown Only</th>
<th>Converted Lexicon - Brown/WSJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default tag order (most likely)</td>
<td>94.2</td>
<td>91.2</td>
<td>~91</td>
</tr>
<tr>
<td>Re-ordered tags</td>
<td>N/A</td>
<td>89.3</td>
<td>76.1</td>
</tr>
</tbody>
</table>

Table 7: Brill Tagger Accuracy Given Different Configurations of the Lexicon
D. Advantages of Lexicon Generation

By generating a lexicon for each corpus presented to the system, the part-of-speech tagger has an advantage over using a standard general lexicon because the scope of the tagset is limited to that of the corpus. Take for example a corpus containing current events, this corpus will use a set of terms in a completely different way than a medical or technical corpus. By limiting the scope of the lexicon to only the instances of terms appearing in the given corpus there is less chance of assigning them the wrong tag.

E. Evaluating Tagged Output Accuracies

To test this hypothesis (H2) we refer to Pereira [26] who used a corpus comprised of news articles randomly selected from Yahoo! News. The articles represented the topic of the day, containing a range of documents, from headline news, to science and technology, to finance, to entertainment, etc. The corpus had a vocabulary of 3,252 terms, but only 1,875 of those terms were found in the Penn-TreeBank lexicon. Phrased another way, the Penn-TreeBank lexicon contains 57.7% of the terms in this corpus. Klas Prütz [28] has demonstrated that the accuracy of the Brill tagger is significantly reduced when run with a lexicon that is missing terms.

By taking the 58% completeness rate as a baseline, an experiment was devised to simulate the imperfect conditions using the Brown Corpus and the Penn-TreeBank lexicon. The experiment has two basic foundations, first, the Penn-TreeBank lexicon contains 100% of the terms contained in the Brown Corpus, and second, the correct answer for the Brown Corpus is available for calculating accuracy measures. By removing 42% of the Penn-TreeBank lexicon, a simulation of the Yahoo! Corpus can be performed on the Brown Corpus. But, since the removal of certain key terms can have a significant impact on the results, this experiment was cross-validated, taking ten result sets into consideration.

The procedure for removing 42% of the Brown Corpus was trivial – ten subset lexicons were created by randomly selecting 58% of the terms contained in the full lexicon and throwing out the remaining terms. Once the ten lexicons (IV2-general-purpose) were created, they were each run through the Brill tagger with the Brown Corpus and then the results were compared against the correct answer to measure its accuracy. The result of the 10-fold validation was an accuracy of 60.1%. A table showing the results of each individual test is shown in Table 8.

<table>
<thead>
<tr>
<th>Run</th>
<th>Percentage</th>
<th>Lexicon Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63.3%</td>
<td>31371</td>
</tr>
<tr>
<td>2</td>
<td>58.7%</td>
<td>31159</td>
</tr>
<tr>
<td>3</td>
<td>56.1%</td>
<td>31252</td>
</tr>
<tr>
<td>4</td>
<td>57.2%</td>
<td>31164</td>
</tr>
<tr>
<td>5</td>
<td>62.3%</td>
<td>31105</td>
</tr>
<tr>
<td>6</td>
<td>61.9%</td>
<td>31364</td>
</tr>
<tr>
<td>7</td>
<td>62.7%</td>
<td>31307</td>
</tr>
<tr>
<td>8</td>
<td>64.7%</td>
<td>31255</td>
</tr>
<tr>
<td>9</td>
<td>53.5%</td>
<td>31239</td>
</tr>
<tr>
<td>10</td>
<td>60.7%</td>
<td>31262</td>
</tr>
</tbody>
</table>

Table 8: 10-Fold Validation and Lexicon Size

F. Discussion

With a resulting accuracy (H2) that is comparable to that of the Penn-TreeBank, the lexicon generation algorithm proposed in this study should continue to be refined. There are a number of improvements that can be made, and with those improvements the accuracy will surely supercede that of a deficient Penn-TreeBank lexicon.

1) Proposed Algorithm Deficiencies

One of the reasons why the proposed algorithm failed to outperform the Penn-TreeBank is because of the heuristic used in step 6 – “grouping terms,” which inefficiently grouped too many terms together. By grouping the terms in a different way, accuracy can be improved. Appendix A shows the full grouping of terms with their correct answers (according to the Penn-TreeBank lexicon). Several groups are correctly identified as being comprised of terms from the same category, for example: group 16 contains mostly terms from category 7; group 28 contains terms primarily from category 4; and group 34 contains terms from category 6. Unfortunately, group 28 is not the only group that contains terms from category 4, group 10 and numerous others also contain words primarily from category 4. Similarly, group 11 shares the properties of group 34, having only words from category 6. Knowing which groups should be combined, like groups 11 and 34, or 10 and 28, is a challenging task. An
attempt to resolve this problem was considered during this research; however, the results were inconclusive and have not been presented. In order to give a general sense of the approach, it is described next.

Once the terms are placed into their groups (i.e. proposed algorithm, step 6), a second pass is made over the result. This second pass looks at all the terms having the same context and iteratively evaluates if a term was mistakenly placed into the wrong group. This evaluation is done by counting how many neighboring terms fall into a grouping different than the current term. If a majority of the neighboring terms fall into a grouping different than the one assigned to the current term, then the term is reassigned to the more frequent grouping. Since the lexicon generation algorithm assigns terms sequentially based on context and morphology, there is a chance that terms are mis-assigned when a term is used in an uncommon context. By re-evaluating a term’s assignment based on its contextual neighbors, the term can be regrouped with terms more likely to be of the same category. This is a time-consuming, and computationally expensive process, which did not produce useful results. A refinement of this algorithm may be worth revisiting in the future. However, it was not pursued for this research due to its unconstrained nature, in other words, it is unclear that this grouping refinement approach will ever be useful.

Still another reason why the proposed algorithm did not perform as well as the Penn-TreeBank lexicon is because of the simplicity employed in step 7 – “assigning categories.” The POS tagger’s sole function is to correctly choose between the options presented in the lexicon in order to tag a document. Unfortunately, the proposed algorithm does not assign terms to multiple categories. The deficiency with step 6, described above, and step 7 go hand-in-hand. If a better method of grouping terms can be devised, then the assignment of categories should become easier, if not part of the grouping itself. With an improved grouping stage that assigns terms to categories along the way, accuracy is expected to surpass that of the Penn-TreeBank.

2) Proposed Algorithm Advantages

Even though the proposed algorithm had some deficiencies, there are several advantages for using and refining such an algorithm. Automatically generating a lexicon from a corpus allows the lexicon to be more precise for the corpus being processed. Corpora of technically dense material, or corpora containing medical terminology, or corpora for general current events will all have different needs when it’s time to process the document. By generating the lexicon, the POS tagging process can begin much sooner, thus reducing the time and resources needed to move on to the next step of processing. Whether it be machine translation, text-to-speech, information retrieval, or a host of other text related activities, with a generated lexicon, the process can start much faster.

V. Conclusion

This paper has provided the background for a common natural language processing problem – part-of-speech tagging; it has presented existing approaches to solve that problem – various POS tagging methods; it then identified a weakness consistent across all of the known solutions – dependency on a previously created lexicon; and it presented a framework to address the weakness – automatic lexicon generation. The framework presented was a method for automatically generating a lexicon to be used in a part-of-speech tagger. Thus, by automatically generating a lexicon, a POS tagger can truly be a self-sufficient, unsupervised, component of larger software. POS taggers can be useful in various linguistic applications including text-to-speech, machine translation, information extraction, and information retrieval. The framework for analysis has been established by introducing a baseline system, which includes the unsupervised Brill tagger, the Penn-TreeBank lexicon, and the Penn-TreeBank tagged Brown Corpus. A detailed discussion of the transformations made to accommodate an automatically generated lexicon was presented and compared with the baseline system. Experimental results show that a generated lexicon is capable of producing competitive results against a general lexicon that is incomplete. Due to misspellings, new words, technical terms, other languages, etc. a general lexicon is not applicable to every corpus, however, a generated lexicon can over-come those deficiencies and address each corpus individually, producing a comparable result. Further research and enhancements to the framework and the algorithm should be pursued to achieve better results.

A. Future Work

Future work on this topic can involve algorithm redevelopment for the “grouping terms” (Step 6) and “assigning categories” (Step 7) of the framework, as described in the discussion section. Additionally, more efficient ways to collect the n-graphs and the left and right contexts can be employed. Currently, these two steps are memory and CPU-intensive and must be run as individual processes. However, with an efficient mechanism, these processes could reuse each other’s efforts to reduce the burden on machine resources. More interestingly, this framework should be applied, and its performance evaluated, on an array of languages other than English. Both the framework and algorithms have been conceived, and proposed, so they can be run on any language without modification. Once it is run on languages other than English, modifications may become necessary based on the findings for each language. With this added experience, the framework and algorithm can be improved to handle languages in a more general sense. Along similar lines, this framework can be applied to specific types of corpora in English, for example, highly technical corpora, or temporally sensitive corpora such as news, or even improper English such as instant messaging. The results of such experiments will add to the justification that this
framework is capable of working on any kind of language, formal or informal, without any external knowledge. The sole requirement is the corpora itself.
REFERENCES


APPENDIX A

Results of the Lexicon Generation Algorithm – Step 6 – “Grouping Terms” without any heuristic imposed. Each word is followed by its correct lexicon entry based on the Penn-TreeBank Lexicon.

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>circulate</td>
<td>7</td>
</tr>
<tr>
<td>precitate</td>
<td>4</td>
</tr>
<tr>
<td>assort</td>
<td>3</td>
</tr>
<tr>
<td>celebrated</td>
<td>3</td>
</tr>
<tr>
<td>feted</td>
<td>3</td>
</tr>
<tr>
<td>text</td>
<td>3</td>
</tr>
<tr>
<td>look-up</td>
<td>4</td>
</tr>
<tr>
<td>disappointed</td>
<td>3</td>
</tr>
<tr>
<td>batted</td>
<td>7</td>
</tr>
<tr>
<td>steal</td>
<td>7</td>
</tr>
<tr>
<td>outstate</td>
<td>3</td>
</tr>
<tr>
<td>smummate</td>
<td>4</td>
</tr>
<tr>
<td>impacted</td>
<td>7</td>
</tr>
<tr>
<td>facilitated</td>
<td>7</td>
</tr>
<tr>
<td>sweated</td>
<td>7</td>
</tr>
<tr>
<td>Cities</td>
<td>7</td>
</tr>
<tr>
<td>maladjusted</td>
<td>3</td>
</tr>
<tr>
<td>eluates</td>
<td>4</td>
</tr>
<tr>
<td>restates</td>
<td>7</td>
</tr>
<tr>
<td>meteor</td>
<td>4</td>
</tr>
<tr>
<td>dasted</td>
<td>7</td>
</tr>
<tr>
<td>Quartet</td>
<td>4</td>
</tr>
<tr>
<td>benefited</td>
<td>7</td>
</tr>
<tr>
<td>Leyte</td>
<td>4</td>
</tr>
<tr>
<td>Debutante</td>
<td>4</td>
</tr>
<tr>
<td>polyphosphate</td>
<td>4</td>
</tr>
<tr>
<td>Completed</td>
<td>7</td>
</tr>
<tr>
<td>tasted</td>
<td>7</td>
</tr>
<tr>
<td>textbook</td>
<td>4</td>
</tr>
<tr>
<td>ejected</td>
<td>7</td>
</tr>
<tr>
<td>catapulated</td>
<td>7</td>
</tr>
<tr>
<td>Industrialistes</td>
<td>4</td>
</tr>
<tr>
<td>compose</td>
<td>4</td>
</tr>
<tr>
<td>composites</td>
<td>4</td>
</tr>
<tr>
<td>Stalinist-corrupted</td>
<td>3</td>
</tr>
<tr>
<td>resisted</td>
<td>7</td>
</tr>
<tr>
<td>test-run</td>
<td>4</td>
</tr>
<tr>
<td>decorate</td>
<td>7</td>
</tr>
<tr>
<td>reputedly</td>
<td>6</td>
</tr>
<tr>
<td>Cartesian</td>
<td>3</td>
</tr>
<tr>
<td>playmate</td>
<td>4</td>
</tr>
<tr>
<td>Societe</td>
<td>4</td>
</tr>
<tr>
<td>designates</td>
<td>7</td>
</tr>
<tr>
<td>committed</td>
<td>7</td>
</tr>
<tr>
<td>step-by-step</td>
<td>3</td>
</tr>
<tr>
<td>undetectable</td>
<td>3</td>
</tr>
<tr>
<td>repeatedly</td>
<td>6</td>
</tr>
<tr>
<td>Russian-dominated</td>
<td>3</td>
</tr>
<tr>
<td>Buffeted</td>
<td>7</td>
</tr>
<tr>
<td>Fate</td>
<td>4</td>
</tr>
<tr>
<td>carriage-step</td>
<td>4</td>
</tr>
<tr>
<td>decorated</td>
<td>3</td>
</tr>
<tr>
<td>spiritual</td>
<td>3</td>
</tr>
<tr>
<td>rosettes</td>
<td>4</td>
</tr>
<tr>
<td>Israeliite</td>
<td>4</td>
</tr>
<tr>
<td>concocted</td>
<td>7</td>
</tr>
<tr>
<td>chatte</td>
<td>2</td>
</tr>
<tr>
<td>Proximate</td>
<td>3</td>
</tr>
<tr>
<td>illustrates</td>
<td>7</td>
</tr>
<tr>
<td>motet</td>
<td>4</td>
</tr>
<tr>
<td>extracted</td>
<td>7</td>
</tr>
<tr>
<td>exacted</td>
<td>7</td>
</tr>
<tr>
<td>on-site</td>
<td>3</td>
</tr>
<tr>
<td>fermented</td>
<td>7</td>
</tr>
<tr>
<td>hotel</td>
<td>4</td>
</tr>
<tr>
<td>devasted</td>
<td>7</td>
</tr>
<tr>
<td>uncharted</td>
<td>3</td>
</tr>
<tr>
<td>lightest</td>
<td>3</td>
</tr>
<tr>
<td>liquidated</td>
<td>7</td>
</tr>
<tr>
<td>misquoted</td>
<td>7</td>
</tr>
<tr>
<td>Anticipated</td>
<td>7</td>
</tr>
<tr>
<td>intoxicated</td>
<td>3</td>
</tr>
<tr>
<td>Fizirte</td>
<td>4</td>
</tr>
<tr>
<td>Bible-emanipulated</td>
<td>3</td>
</tr>
<tr>
<td>obsolete</td>
<td>3</td>
</tr>
<tr>
<td>unassisted</td>
<td>3</td>
</tr>
<tr>
<td>Stems</td>
<td>4</td>
</tr>
<tr>
<td>shouted</td>
<td>7</td>
</tr>
<tr>
<td>humiliated</td>
<td>7</td>
</tr>
<tr>
<td>simulated</td>
<td>7</td>
</tr>
<tr>
<td>note</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>self-dictrate</td>
<td>4</td>
</tr>
<tr>
<td>retaliated</td>
<td>7</td>
</tr>
<tr>
<td>Stewart</td>
<td>4</td>
</tr>
<tr>
<td>steppers</td>
<td>4</td>
</tr>
<tr>
<td>d'Hostel</td>
<td>2</td>
</tr>
<tr>
<td>haute</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>fretted</td>
<td>7</td>
</tr>
<tr>
<td>unstitled</td>
<td>3</td>
</tr>
<tr>
<td>tea</td>
<td>4</td>
</tr>
<tr>
<td>Item-Categories</td>
<td>4</td>
</tr>
<tr>
<td>hooted</td>
<td>7</td>
</tr>
<tr>
<td>Subsystems</td>
<td>4</td>
</tr>
<tr>
<td>teleology</td>
<td>4</td>
</tr>
<tr>
<td>amounted</td>
<td>7</td>
</tr>
<tr>
<td>implanted</td>
<td>7</td>
</tr>
<tr>
<td>half-educated</td>
<td>3</td>
</tr>
<tr>
<td>best-hearted</td>
<td>3</td>
</tr>
<tr>
<td>newly-appointed</td>
<td>3</td>
</tr>
<tr>
<td>mates</td>
<td>4</td>
</tr>
<tr>
<td>wrote</td>
<td>7</td>
</tr>
<tr>
<td>create</td>
<td>7</td>
</tr>
<tr>
<td>stews</td>
<td>4</td>
</tr>
<tr>
<td>items</td>
<td>4</td>
</tr>
<tr>
<td>Butte</td>
<td>4</td>
</tr>
<tr>
<td>benighted</td>
<td>3</td>
</tr>
<tr>
<td>gate-post</td>
<td>4</td>
</tr>
<tr>
<td>acquitted</td>
<td>7</td>
</tr>
<tr>
<td>constituted</td>
<td>7</td>
</tr>
<tr>
<td>Australies</td>
<td>4</td>
</tr>
<tr>
<td>Sextet</td>
<td>4</td>
</tr>
<tr>
<td>steadfastly</td>
<td>6</td>
</tr>
<tr>
<td>slotted</td>
<td>7</td>
</tr>
<tr>
<td>Despite</td>
<td>1</td>
</tr>
<tr>
<td>technical</td>
<td>3</td>
</tr>
<tr>
<td>Fortescue</td>
<td>4</td>
</tr>
<tr>
<td>hates</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Revisited</td>
<td>4</td>
</tr>
<tr>
<td>keynote</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>consolidated</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>nearingly</td>
<td>6</td>
</tr>
<tr>
<td>carbontes</td>
<td>4</td>
</tr>
<tr>
<td>Primate</td>
<td>4</td>
</tr>
<tr>
<td>coveted</td>
<td>7</td>
</tr>
<tr>
<td>halfhearted</td>
<td>3</td>
</tr>
<tr>
<td>disaffiliated</td>
<td>3</td>
</tr>
<tr>
<td>Orthega</td>
<td>4</td>
</tr>
<tr>
<td>silicates</td>
<td>4</td>
</tr>
<tr>
<td>footsteps</td>
<td>4</td>
</tr>
<tr>
<td>recruited</td>
<td>7</td>
</tr>
<tr>
<td>sweet-throated</td>
<td>3</td>
</tr>
<tr>
<td>Inmates</td>
<td>4</td>
</tr>
<tr>
<td>directorate</td>
<td>4</td>
</tr>
<tr>
<td>iodinated</td>
<td>7</td>
</tr>
<tr>
<td>Rates</td>
<td>4</td>
</tr>
<tr>
<td>detector</td>
<td>4</td>
</tr>
<tr>
<td>violate</td>
<td>7</td>
</tr>
<tr>
<td>private</td>
<td>3</td>
</tr>
<tr>
<td>slatted</td>
<td>3</td>
</tr>
<tr>
<td>silicate</td>
<td>4</td>
</tr>
<tr>
<td>regulated</td>
<td>7</td>
</tr>
<tr>
<td>Plate</td>
<td>4</td>
</tr>
<tr>
<td>uncounted</td>
<td>3</td>
</tr>
<tr>
<td>State</td>
<td>4</td>
</tr>
<tr>
<td>cultivated</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>irate</td>
<td>3</td>
</tr>
<tr>
<td>Tastes</td>
<td>4</td>
</tr>
<tr>
<td>telesological</td>
<td>3</td>
</tr>
<tr>
<td>teaspoon</td>
<td>4</td>
</tr>
<tr>
<td>Satellites</td>
<td>4</td>
</tr>
<tr>
<td>neatest</td>
<td>3</td>
</tr>
<tr>
<td>distracted</td>
<td>7</td>
</tr>
<tr>
<td>Step</td>
<td>4</td>
</tr>
<tr>
<td>recreated</td>
<td>7</td>
</tr>
<tr>
<td>Rockettes</td>
<td>4</td>
</tr>
<tr>
<td>inundated</td>
<td>3</td>
</tr>
<tr>
<td>graduated</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>salivate</td>
<td>7</td>
</tr>
<tr>
<td>bolted</td>
<td>7</td>
</tr>
<tr>
<td>variegated</td>
<td>3</td>
</tr>
<tr>
<td>visited</td>
<td>7</td>
</tr>
<tr>
<td>ignite</td>
<td>7</td>
</tr>
<tr>
<td>exhibited</td>
<td>7</td>
</tr>
<tr>
<td>plated</td>
<td>7</td>
</tr>
<tr>
<td>unlimited</td>
<td>3</td>
</tr>
<tr>
<td>sweetest</td>
<td>3</td>
</tr>
<tr>
<td>substituted</td>
<td>7</td>
</tr>
<tr>
<td>Immediately</td>
<td>6</td>
</tr>
<tr>
<td>redecorated</td>
<td>7</td>
</tr>
<tr>
<td>acutely</td>
<td>6</td>
</tr>
<tr>
<td>effectuate</td>
<td>7</td>
</tr>
<tr>
<td>Fichte</td>
<td>4</td>
</tr>
<tr>
<td>deflated</td>
<td>3</td>
</tr>
<tr>
<td>juste</td>
<td>2</td>
</tr>
<tr>
<td>teddy</td>
<td>4</td>
</tr>
<tr>
<td>delicately</td>
<td>6</td>
</tr>
<tr>
<td>crystallites</td>
<td>4</td>
</tr>
<tr>
<td>blanketed</td>
<td>7</td>
</tr>
<tr>
<td>Steams</td>
<td>4</td>
</tr>
<tr>
<td>averted</td>
<td>7</td>
</tr>
<tr>
<td>confiscated</td>
<td>7</td>
</tr>
<tr>
<td>redistributed</td>
<td>7</td>
</tr>
<tr>
<td>Devoted</td>
<td>7</td>
</tr>
<tr>
<td>pretext</td>
<td>4</td>
</tr>
<tr>
<td>Breasted</td>
<td>4</td>
</tr>
<tr>
<td>hand-painted</td>
<td>4</td>
</tr>
<tr>
<td>granite</td>
<td>4</td>
</tr>
<tr>
<td>Mattei</td>
<td>4</td>
</tr>
<tr>
<td>purportedly</td>
<td>6</td>
</tr>
<tr>
<td>tells</td>
<td>7</td>
</tr>
<tr>
<td>pupates</td>
<td>7</td>
</tr>
<tr>
<td>United</td>
<td>4</td>
</tr>
<tr>
<td>invested</td>
<td>7</td>
</tr>
<tr>
<td>quoted</td>
<td>7</td>
</tr>
<tr>
<td>Steel</td>
<td>4</td>
</tr>
<tr>
<td>distributes</td>
<td>7</td>
</tr>
<tr>
<td>perpetrated</td>
<td>7</td>
</tr>
<tr>
<td>alkylarsonate</td>
<td>4</td>
</tr>
<tr>
<td>Limited</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>enacted</td>
<td>7</td>
</tr>
<tr>
<td>astute</td>
<td>3</td>
</tr>
<tr>
<td>contrasted</td>
<td>7</td>
</tr>
<tr>
<td>categorical</td>
<td>3</td>
</tr>
<tr>
<td>Keynotes</td>
<td>4</td>
</tr>
<tr>
<td>bunkmate</td>
<td>4</td>
</tr>
<tr>
<td>slatted</td>
<td>3</td>
</tr>
<tr>
<td>silicate</td>
<td>4</td>
</tr>
<tr>
<td>regulate</td>
<td>7</td>
</tr>
<tr>
<td>Dissented</td>
<td>7</td>
</tr>
<tr>
<td>university-educated</td>
<td>3</td>
</tr>
<tr>
<td>approximately</td>
<td>6</td>
</tr>
<tr>
<td>enetered</td>
<td>7</td>
</tr>
<tr>
<td>votes</td>
<td>4</td>
</tr>
<tr>
<td>Protestant-dominated</td>
<td>3</td>
</tr>
<tr>
<td>unenunciated</td>
<td>3</td>
</tr>
<tr>
<td>blasted</td>
<td>7</td>
</tr>
<tr>
<td>polie</td>
<td>3</td>
</tr>
<tr>
<td>allocate</td>
<td>7</td>
</tr>
<tr>
<td>cutest</td>
<td>4</td>
</tr>
<tr>
<td>carpeted</td>
<td>7</td>
</tr>
<tr>
<td>trans-illuminated</td>
<td>3</td>
</tr>
<tr>
<td>stimulated</td>
<td>7</td>
</tr>
<tr>
<td>repudiated</td>
<td>7</td>
</tr>
<tr>
<td>Elite</td>
<td>4</td>
</tr>
<tr>
<td>desegregate</td>
<td>7</td>
</tr>
<tr>
<td>persisted</td>
<td>7</td>
</tr>
<tr>
<td>pig-infested</td>
<td>3</td>
</tr>
<tr>
<td>mate</td>
<td>4</td>
</tr>
<tr>
<td>eluted</td>
<td>7</td>
</tr>
<tr>
<td>Cassie</td>
<td>4</td>
</tr>
<tr>
<td>Brunettes</td>
<td>4</td>
</tr>
<tr>
<td>impersonated</td>
<td>7</td>
</tr>
<tr>
<td>exemptorize</td>
<td>7</td>
</tr>
<tr>
<td>fites</td>
<td>4</td>
</tr>
<tr>
<td>Plate</td>
<td>4</td>
</tr>
<tr>
<td>etes</td>
<td>2</td>
</tr>
<tr>
<td>Systemic</td>
<td>3</td>
</tr>
<tr>
<td>tainted</td>
<td>7</td>
</tr>
<tr>
<td>unventilated</td>
<td>7</td>
</tr>
<tr>
<td>immediate</td>
<td>3</td>
</tr>
<tr>
<td>Redoute</td>
<td>4</td>
</tr>
<tr>
<td>hesitated</td>
<td>7</td>
</tr>
<tr>
<td>adjudicate</td>
<td>7</td>
</tr>
<tr>
<td>drafted</td>
<td>7</td>
</tr>
<tr>
<td>Estate</td>
<td>4</td>
</tr>
<tr>
<td>platted</td>
<td>7</td>
</tr>
<tr>
<td>technology</td>
<td>4</td>
</tr>
<tr>
<td>repropate</td>
<td>4</td>
</tr>
<tr>
<td>layette</td>
<td>4</td>
</tr>
<tr>
<td>attracted</td>
<td>7</td>
</tr>
<tr>
<td>schoolmate</td>
<td>4</td>
</tr>
<tr>
<td>double-breasted</td>
<td>3</td>
</tr>
<tr>
<td>Arte</td>
<td>4</td>
</tr>
<tr>
<td>warmedhearted</td>
<td>3</td>
</tr>
<tr>
<td>pro-tem</td>
<td>3</td>
</tr>
<tr>
<td>consummated</td>
<td>7</td>
</tr>
<tr>
<td>assisted</td>
<td>7</td>
</tr>
<tr>
<td>steam-baths</td>
<td>4</td>
</tr>
<tr>
<td>Separate</td>
<td>3</td>
</tr>
<tr>
<td>states</td>
<td>4</td>
</tr>
<tr>
<td>allotted</td>
<td>7</td>
</tr>
<tr>
<td>Noted</td>
<td>3</td>
</tr>
<tr>
<td>Compute</td>
<td>7</td>
</tr>
<tr>
<td>hotelman</td>
<td>4</td>
</tr>
<tr>
<td>Monte</td>
<td>4</td>
</tr>
<tr>
<td>anticipated</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>discounted</td>
<td>7</td>
</tr>
<tr>
<td>heretofore-accepted</td>
<td>3</td>
</tr>
<tr>
<td>barefooted</td>
<td>3</td>
</tr>
<tr>
<td>Strategy</td>
<td>4</td>
</tr>
<tr>
<td>Estep</td>
<td>4</td>
</tr>
<tr>
<td>flouted</td>
<td>7</td>
</tr>
<tr>
<td>gates</td>
<td>4</td>
</tr>
<tr>
<td>closeted</td>
<td>7</td>
</tr>
<tr>
<td>Committee</td>
<td>4</td>
</tr>
<tr>
<td>prompted</td>
<td>7</td>
</tr>
<tr>
<td>unites</td>
<td>7</td>
</tr>
<tr>
<td>irritates</td>
<td>7</td>
</tr>
<tr>
<td>lately</td>
<td>6</td>
</tr>
<tr>
<td>palates</td>
<td>4</td>
</tr>
<tr>
<td>blared</td>
<td>7</td>
</tr>
<tr>
<td>Guatemala</td>
<td>4</td>
</tr>
<tr>
<td>sophisticated</td>
<td>3</td>
</tr>
<tr>
<td>Latest</td>
<td>3</td>
</tr>
<tr>
<td>contented</td>
<td>3</td>
</tr>
<tr>
<td>reacted</td>
<td>7</td>
</tr>
<tr>
<td>Seated</td>
<td>7</td>
</tr>
<tr>
<td>Gotrey</td>
<td>4</td>
</tr>
<tr>
<td>dissipated</td>
<td>7</td>
</tr>
<tr>
<td>Note</td>
<td>7</td>
</tr>
<tr>
<td>illustrated</td>
<td>7</td>
</tr>
<tr>
<td>satellite</td>
<td>3</td>
</tr>
<tr>
<td>Cigarette</td>
<td>4</td>
</tr>
<tr>
<td>united</td>
<td>3</td>
</tr>
<tr>
<td>originated</td>
<td>3</td>
</tr>
<tr>
<td>accumultes</td>
<td>7</td>
</tr>
<tr>
<td>importantly</td>
<td>6</td>
</tr>
<tr>
<td>Descarates</td>
<td>4</td>
</tr>
<tr>
<td>halted</td>
<td>7</td>
</tr>
<tr>
<td>abstracted</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>phosphates</td>
<td>4</td>
</tr>
<tr>
<td>frustrate</td>
<td>7</td>
</tr>
<tr>
<td>wilted</td>
<td>3</td>
</tr>
<tr>
<td>tabulated</td>
<td>3</td>
</tr>
<tr>
<td>bare-footed</td>
<td>3</td>
</tr>
<tr>
<td>immediate</td>
<td>3</td>
</tr>
<tr>
<td>dynamitted</td>
<td>7</td>
</tr>
<tr>
<td>Redoute</td>
<td>4</td>
</tr>
<tr>
<td>hesitated</td>
<td>7</td>
</tr>
<tr>
<td>adjudicate</td>
<td>7</td>
</tr>
<tr>
<td>drafted</td>
<td>7</td>
</tr>
<tr>
<td>Macwhyte</td>
<td>4</td>
</tr>
<tr>
<td>Frosbite</td>
<td>4</td>
</tr>
<tr>
<td>repropate</td>
<td>4</td>
</tr>
<tr>
<td>daunited</td>
<td>7</td>
</tr>
<tr>
<td>estates</td>
<td>4</td>
</tr>
<tr>
<td>flaggedell</td>
<td>7</td>
</tr>
<tr>
<td>fronted</td>
<td>7</td>
</tr>
<tr>
<td>footnote</td>
<td>4</td>
</tr>
<tr>
<td>architecture</td>
<td>4</td>
</tr>
<tr>
<td>drafees</td>
<td>4</td>
</tr>
<tr>
<td>Campitelli</td>
<td>4</td>
</tr>
<tr>
<td>Uto-Aztec</td>
<td>4</td>
</tr>
<tr>
<td>candidates</td>
<td>4</td>
</tr>
<tr>
<td>elevated</td>
<td>3</td>
</tr>
<tr>
<td>evaluate</td>
<td>7</td>
</tr>
<tr>
<td>dedicates</td>
<td>7</td>
</tr>
<tr>
<td>neon-lighted</td>
<td>3</td>
</tr>
<tr>
<td>sprinted</td>
<td>7</td>
</tr>
<tr>
<td>reacquainted</td>
<td>7</td>
</tr>
<tr>
<td>telescopic</td>
<td>3</td>
</tr>
<tr>
<td>discoverer 4</td>
<td>worsen 7</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>fasten 7</td>
<td>Monday 4</td>
</tr>
<tr>
<td>Persia 4</td>
<td>skyjacked 7</td>
</tr>
<tr>
<td>maneuvered 7</td>
<td>Connelly 4</td>
</tr>
<tr>
<td>Leaders 4</td>
<td>Lagerlof 4</td>
</tr>
<tr>
<td>one-plane 3</td>
<td>nondescriptly 6</td>
</tr>
<tr>
<td>tampered 7</td>
<td>shallower 3</td>
</tr>
<tr>
<td>otter 4</td>
<td>millivolter 4</td>
</tr>
<tr>
<td>Oversized 3</td>
<td>kedgeree 4</td>
</tr>
<tr>
<td>self-employed 3</td>
<td>Britisher 4</td>
</tr>
<tr>
<td>cito 2</td>
<td>Contrary 3</td>
</tr>
<tr>
<td>Colonel 4</td>
<td>saged 7</td>
</tr>
<tr>
<td>founders 4</td>
<td>leverage 4</td>
</tr>
<tr>
<td>long-life 3</td>
<td>moderator 4</td>
</tr>
<tr>
<td>paragon 4</td>
<td>imperialist 4</td>
</tr>
<tr>
<td>Emmerich 4</td>
<td>exploiters 4</td>
</tr>
<tr>
<td>ponds 4</td>
<td>overcome 7</td>
</tr>
<tr>
<td>Water 4</td>
<td>hidden 3</td>
</tr>
<tr>
<td>accompanied 7</td>
<td>Leibowitz 4</td>
</tr>
<tr>
<td>aerals 4</td>
<td>fumed 7</td>
</tr>
<tr>
<td>Densmore 4</td>
<td>Rattzenflut 4</td>
</tr>
<tr>
<td>Othon 4</td>
<td>typewriter 4</td>
</tr>
<tr>
<td>ex-convicts 4</td>
<td>kidnappers 4</td>
</tr>
<tr>
<td>daughters 4</td>
<td>non-readers 4</td>
</tr>
<tr>
<td>electronics 4</td>
<td>dammit 1</td>
</tr>
<tr>
<td>Universal 4</td>
<td>flattered 7</td>
</tr>
<tr>
<td>hovering 7</td>
<td>well-played 3</td>
</tr>
<tr>
<td>Land-Rover 4</td>
<td>statesmen 4</td>
</tr>
<tr>
<td>calligraphers 4</td>
<td>enforces 7</td>
</tr>
<tr>
<td>nonsensical 3</td>
<td>Greenwhich 4</td>
</tr>
<tr>
<td>Wheeler 4</td>
<td>Arteries 4</td>
</tr>
<tr>
<td>spliced 7</td>
<td>Bouvardier 4</td>
</tr>
<tr>
<td>armpits 4</td>
<td>Wissachicken 4</td>
</tr>
<tr>
<td>exuberantly 6</td>
<td>Lasswitz 4</td>
</tr>
<tr>
<td>hardboiled 3</td>
<td>Marenozio 4</td>
</tr>
<tr>
<td>Saloon 4</td>
<td>dissonances 4</td>
</tr>
<tr>
<td>Constant 3</td>
<td>recovered 7</td>
</tr>
<tr>
<td>retired 3</td>
<td>Balenciaga 4</td>
</tr>
<tr>
<td>tissue-organised 3</td>
<td>Recovery 4</td>
</tr>
<tr>
<td>Bouvardier 4</td>
<td>spliced 7</td>
</tr>
<tr>
<td>Arteries 4</td>
<td>nonsensical 3</td>
</tr>
<tr>
<td>enforced 7</td>
<td>calligraphers 4</td>
</tr>
<tr>
<td>statesmen 4</td>
<td>Land-Rover 4</td>
</tr>
<tr>
<td>hovered 7</td>
<td>fluttered 7</td>
</tr>
<tr>
<td>Universal 4</td>
<td>electronics 4</td>
</tr>
<tr>
<td>non-readers 4</td>
<td>typewriter 4</td>
</tr>
<tr>
<td>Othon 4</td>
<td>Rattzhenfuut 4</td>
</tr>
<tr>
<td>Densmore 4</td>
<td>fumed 7</td>
</tr>
<tr>
<td>aerials 4</td>
<td>lounged 7</td>
</tr>
<tr>
<td>Reifenrath 4</td>
<td>terror-stricken 3</td>
</tr>
<tr>
<td>Required 7</td>
<td>modernists 4</td>
</tr>
<tr>
<td>Peterson 4</td>
<td>stickler 4</td>
</tr>
<tr>
<td>十余 4</td>
<td>populous-sponsored 3</td>
</tr>
<tr>
<td>gliders 3</td>
<td>decelerate 7</td>
</tr>
<tr>
<td>landowners 4</td>
<td>half-turned 3</td>
</tr>
<tr>
<td>Bluthenweig 4</td>
<td>tactly 6</td>
</tr>
<tr>
<td>disarranged 7</td>
<td>Wilfred 4</td>
</tr>
<tr>
<td>expense 4</td>
<td>So-called 4</td>
</tr>
<tr>
<td>carbon-14 4</td>
<td>helped 7</td>
</tr>
<tr>
<td>hunter 4</td>
<td>songs 4</td>
</tr>
<tr>
<td>chaffed 3</td>
<td>deluded 3</td>
</tr>
<tr>
<td>been 7</td>
<td>Photographer 4</td>
</tr>
<tr>
<td>stickler 4</td>
<td>Arbeitskommando 4</td>
</tr>
<tr>
<td>modernists 4</td>
<td>ecumenicists 4</td>
</tr>
<tr>
<td>Required 7</td>
<td>terror-stricken 3</td>
</tr>
<tr>
<td>Reifenrath 4</td>
<td>now-misplaced 3</td>
</tr>
</tbody>
</table>
grinning 3 7
hell-raising 4
criss-crossing 4
clicking 7
severe-looking 3
three-inning 3
totaling 7
re-thinking 7
gaping 7
Goering 4
expounding 7
mushrooming 7
woodworking 4
hijacking 4 7
indexing 4 7
transcending 7
cotton-growing 3
Sparling 4
gazing 4
marring 7
shaking 7
seafaring 3
electrifying 3
baking 4 7
delving 3
marshalling 4
uncompromising 3
Washing 4
zigzagging 7
tracking 7
nagging 3 4 7
commanding 3 7
Thinking 4 7
preening 7
programming 4
fingering 7
canning 4 7
knot-tying 3
scaffolding 4
comparing 7
spinning 7
sinning 7
Travelling 3
egg-hatching 3
taxing 3 4 7
occupying 3 7
mudging 7
nursing 7
naming 4 7
scissoring 7
cornering 7
gorging 7
railing 4
upholding 4
vouching 7
chopping 4 7
Messing 7
sacrificing 7
settling 7
Nearing 7
puddling 7
glazing 7
dodging 7
spear-throwing 3
telescoping 4
handling 4 7
Joking 7
sunning 7
Drawing 4 7
Wondering 7
Commandeer 7
tinkering 4 7
Trying 4
whistling 7
blacking 4
gambling 4 7
accomplishing 7
Lessing 4
strengthening 4 7
swooping 7
String 7
unreasoning 3
prodding 7
clarifying 7
fostering 4 7
drinking 4 7
Harding 4
Telling 7
fast-firing 3
marching 4 7
calving 4 7
swallowing 7
causing 3 7
harassing 7
devising 7
loving 3 7
riddling 7
Linking 7
unceasing 3
Moving 7
Court-pack 3
engulfing 3
ruining 7
textile-producing 3
recovering 7
warming 7
smuggling 7
peeling 4 7
itemizing 7
running 7
complaining 7
soldiering 4
towering 3 7
muscle-shaping 3
providing 7
Shipbuilding 4
recurring 7
snoring 4 7
Manufacturing 4
appealing 3 7
mobilizing 4 7
pseudo-willing 4
lurching 7
bettering 7
Owing 6
living 4 7
enduring 4
honoring 7
coiling 7
draining 4 7
sassing 4
circling 7
perplexing 3
pausing 7
eye-gouging 4
effectuating 3
carloading 4
tensioning 7
unblushing 3
reтелing 4
drawing 4 7
Understanding 4 7
slow-scrambling 3
depriving 7
ever-increasing 3
Stammering 4
stooping 7
dishonouring 7
thumping 7
horse-trading 4
good-looking 3
firing 3 4 7
vanishing 7
slackening 7
onrushing 3
depending 7
Changing 3 7
refusing 7
passing 4 7
unplacing 7
scudding 7
upstanding 3
suffering 4 7
endearing 3
Homecoming 4
stabilizing-conserving 3
selling 3 4 7
modeling 3
ballooning 7
belaboring 4 7
conceeding 7
offing 4
packaging 4 7
canvassing 7
edging 4 7
shearing 4 7
longing 4 7
scratchy 7
crowing 4 7
Ruling 4 7
bathing 4
ripping 3
shortening 7
flapping 3 7
bucking 7
appetizing 3
raiding 7
inning 4
Mountaineering 4
browning 7
varying 7
discouraging 3 7
Hang 7
eye-filling 3
door-to-floor 3
installing 7
crusading 7
purpling 7
important-looking 3
whipping 3 4 7
bucking 7
 cried 7
Kneeling 7
undying 3
easy-going 3
Applying 7
Mining 4
ever-growing 3
pitching 3
dooming 7
seeding 7
qualifying 7
Dropping 7
tall-growing 3
facing 7
Thanksgiving 4
grieving 7
pumping 7
sparking 3 7
changing 7
Laughing 4
v pitying 3
Hastening 7
imploring 7
reasoning 4
overpowering 3
sky-carving 3
tracing 4 7
materials-handling 3
deliberately 7
thinking 4 7
musing 7
engineering 7
examining 4 7
napping 7
Burning 4
Bandaging 4
discharging 7
figuring 7
refreshing 3
belonging 7
reviewing 7
Squeezing 7
boring 3 4 7
self-pitying 7
Varying 7
non-bearing 3
Matching 3 7
sapping 7
Teaching 7
whining 7
persuing 7
requiring 7
muster ing 7
angling 7
Peking 4
tollering 3 7
striving 4 7
trampling 4
careless 4 7
involving 7
sketching 4 7
Awakening 4
prying 7
Staining 7
bricklaying 4
motel-keeping 4
recommend ing 7
Carving 4
Referencing 7
paving 7
tugging 7
raking 3 4 7
heat-processing 3
kidnapping 7
nailing 7
law-breaking 4
tidying 7
Singing 4 7
Loving 4
tearing 7
declaring 7
rubbing 7
processing 4 7
enrolling 4
hurrying 3 7
pseudo-thinking 4
partaking 7
Bringing 7
Trailing 4
locking 3 7
Founding 4
Seeking 7
cropping 7
gut-flattening 3
practising 7
propping 7
presiding 7
throwing 7
hindering 3
evidencing 7
grooming 7
time-consuming 3
moving 7
marrying 7
chuffing 7
yodeling 7
dispensing 7
faring 7
ragging 7
promising 3 7
darkling 3
toro-defining 3
foreshortening 7
reviving 7
paraphrasing 7
Weaving 7
Picturing 7
sandig 4 7
switching 4
nervy-shattering 3
wracking 7
logging 4
Swelling 3
economizing 7
Kroening 4
Sport-King 4
all-consuming 3
pseudoc 4
sweet-smelling 3
acknowledging 7
filming 7
viewing 4 7
enjaming 4
shamming 4
shielding 4
refreezing 3
commencing 7
Maximizing 7
sacking 7
waxing 4
searing 7
sticking 7
itching 7
rallying 3 4
pooling 7
Nodding 7
walloping 3
poking 7
dizz ling 3 7
abound 7
abuse 4
absolution 4
accumulation 4
achievement 4
accretion 4
acquisition 4
acidification 4
acidification 4
acidification 4
acidification 4
acquittal 4
acquittal 4
acquittal 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
acquisition 4
Inter-American 4

information-cell 4
information-seeking 4
organization-position 3
radiation-produced 3
foundation-stone 4
desegregation-from-court-order 4
segregationist 3 4
conservationist 4

andrenas 4