2016

The Theoretical Argument for Disproving Asymptotic Upper-Bounds on the Accuracy of Part-of-Speech Tagging Algorithms: Adopting a Linguistics, Rule-Based Approach

William Foley
University of Mississippi. Sally McDonnell Barksdale Honors College

THE THEORETICAL ARGUMENT FOR DISPROVING ASYMPTOTIC UPPER-BOUNDS ON THE ACCURACY OF PART-OF-SPEECH TAGGING ALGORITHMS: ADOPTING A LINGUISTICS, RULE-BASED APPROACH

By
Will Foley

A thesis submitted to the faculty of the University of Mississippi in partial fulfillment of the requirements of the Sally McDonnell Barksdale Honors College

Oxford, MS
May, 2016

Approved By:

_____
Advisor: Associate Professor of Linguistics,
Dr. Allison Burkette

_____
Reader: Dean of Sally McDonnell Barksdale Honors College,
Dr. Sullivan-Gonzales

_____
Reader: Associate Professor of Philosophy,
Dr. Robert Barnard

_____
Reader: Associate Professor of Computer Science,
Dr. Yixin Chen

Abstract

This paper takes a deep dive into a particular area of the interdisciplinary domain of Computational Linguistics, Part-of-Speech Tagging algorithms.

The author relies primarily on scholarly Computer Science and Linguistics papers to describe previous approaches to this task and the often-hypothesized existence of the asymptotic accuracy rate of around 98%, by which this task is allegedly bound. However, after doing more research into why the accuracy of previous algorithms have behaved in this asymptotic manner, the author identifies valid and empirically-backed reasons why the accuracy of previous approaches do not necessarily reflect any sort of general asymptotic bound on the task of automated Part-of-Speech Tagging. In response, a theoretical argument is proposed to circumvent the shortcomings of previous approaches to this task, which involves abandoning the flawed status-quo of training machine learning algorithms and predictive models on outdated corpora, and instead walks the reader from conception through implementation of a rule-based algorithm with roots in both practical and theoretical Linguistics.

While the resulting algorithm is simply a prototype which cannot be currently verified in achieving a tagging-accuracy rate of over 98%, its multi-tiered methodology, meant to mirror aspects of human cognition in Natural Language Understanding, is meant to serve as a theoretical blueprint for a new and inevitably more-reliable way to deal with the challenges in Part-of-Speech Tagging, and provide much-needed advances in the popular area of Natural Language Processing.

Table of Contents

Part I: Problem Overview and Existing Approaches

Chapter 1: An Introduction to Part of Speech Tagging……....2

Chapter 2: A Soft Introduction to Statistical and Machine Learning…….. 6

Chapter 3: Common Threads in Existing Methodologies……...11

Chapter 4: Various Machine Learning and Mixed Approaches……... 14

Chapter 5: Critique of Former Methodologies: The Need for Linguistics..... 19

Part Two: A New, Linguistic and Rule-Based Approach

Chapter 6: Strategically Defining a Minimal Tagset..... 24

Chapter 7: Establishing Methodology..... 35

Chapter 8: Design Specifications and Implementation..... 39

Chapter 9: Measuring Accuracy and Success..... 44

Bibliography..... 49

Part I:

Problem Overview and Existing Approaches

Chapter 1: An Introduction to Part of Speech Tagging

A Part of Speech (PoS) Tagging Algorithm is a computer program or system that attempts to "tag" words in a given sentence with the grammatical class or word class under which these words, in usage, should technically fall. In the context of this paper and generally in the study of Linguistics, the terms *part of speech, grammatical class, grammatical category, lexical class, lexical category,* and *word class* are used interchangeably. Such an algorithm, as all algorithms, can be broken down by its input and expected output. For this algorithm, its input is simply an unmarked, unannotated sentence. The sentence should be an intelligible one that adheres at least loosely to the norms of Standard English. Otherwise, attempting to identify the parts of speech of its constituent words, whether from a rule-based or statistics-based perspective is not feasible. The last sentence gives insight into the expected output of the algorithm. Given a coherent English sentence, the algorithm seeks to output a sequence of parts of speech (e.g. Noun, Verb, Preposition), often in an abbreviated form (e.g. N, V, PS).

The parts of speech with which the algorithm is already equipped to mark words for their grammatical class is referred to as the algorithm's *tagset*. If one looks across all of the PoS tagging algorithms already in existence, many of their tagsets are of varying length. In other words, the extent to which they seek to distinguish between words' parts of speech varies greatly. The programmer's choice when defining his or her PoS tagging

algorithm in defining the algorithm's tagset is of great importance to the algorithm's accuracy and potential applications. This point will be analyzed in depth later in this paper.

The applications of a good PoS tagging algorithm are of great variety and importance in both practical and computational linguistics, namely within the domain of NLP. For practical use, a linguist performing discourse analysis may search for words or collocates of a word, but only so long as the word being searched for is captured in usage as being one part of speech instead of another part of speech. For example, a linguist researching what kind of subjects perform the action "to fight" would search for collocates, or surrounding words, of the word *fight* over a corpus such as the famous and frequently-used BYU-constructed COCA or COHA in order to gather empirical evidence surrounding this question. However, in her search, she may want to avoid all instances in which *fight* is used as a noun. While this is possible and done frequently when running queries over already-tagged, notorious corpora like COCA and COHA, linguistics are not afforded this luxury when they want to perform similar research on other, perhaps custom-created or personally gathered corpora, such as the transcripts in a Presidential Debate or transcribed and translated manuscripts from field work. Having a highly-accurate PoS tagging algorithm then becomes invaluable to tagging these unannotated corpora so that queries and searches that rely on distinguishing between the grammatical class of certain words is necessary.

In the domain of Natural Language Processing (NLP), the function of an effective PoS tagging algorithm is as equally, if not more, important. In order to process language for its semantics and extract the intended meaning, which require advanced algorithms in

itself, words must first be accurately tagged for their part of speech or grammatical class. As a simple example, Apple's Siri would never be able to give you directions to Disney World if it first processed Disney World as a verb or anything besides a proper noun in the milliseconds after your inquiry into how to get there. For this and a plethora of other reasons, an accurate PoS tagging algorithm, or a way to first deduce the grammatical class of given words, is an indispensable tool for any software engineer or programmer working within the realm of NLP.

So what is so hard about building a 100% accurate PoS tagging algorithm? Deducing a word's part of speech in usage is a trivial and intuitive task for any speaker of English, but attempting to do so algorithmically poses numerous challenges that pervade a number of different approaches that have been taken, including machine-learning based approaches ranging from Hidden Markov Models (HMM's) to Artificial Neural Networks (ANN's) in developing such an algorithm. To truly understand the problems faced by researchers attempting to program a highly-accurate PoS tagging algorithm so that a better one can be constructed, it is necessary to first analyze the various approaches taken by researchers in the past, and identify the reason for their shortcomings. These approaches include predictive, statistical models and Artificial Neural Networks (ANN's) to elementary attempts at Rule-Based PoS Tagging Algorithms, along with the integration and sequential processing of everything in between. But this is not enough. One must also analyze these shortcomings in the nature of the context in which they take place - the science of Linguistics, and the still-raging arguments among linguists regarding the ideal of grammatical classes and the question of how granular of distinctions one may make between parts of speech before the entire concept evaporates

entirely. Additionally, the statistics surrounding the fundamental problem of single words belonging to more than one word class depending on their usage must be closely examined. Only then, after taking all of these things into consideration, may one come to the conclusion that the only hope for an acceptably accurate PoS tagging algorithm, which is capable of actually being deployed by linguists and NLP researchers for practical purposes, must be one with a relatively small tagset and a purely logic-based, rule-driven methodology for correctly applying these tags and identifying the part of speech of an input sentence's words. This is precisely the argument which the author hopes to prove, and the PoS tagging algorithm he hopes to construct.

Chapter 2: A Soft Introduction to Statistical and Machine Learning

As hinted in Chapter 1, Part of Speech (PoS) Tagging algorithms, otherwise known as PoS taggers, have existed and been a point of focus for many researchers in the domains of Computational Linguistics and NLP for quite a while now, with most research and high-accuracy algorithms developed in the last two decades due primarily to the advancements made in machine learning, or computational statistical learning methods. Before diving deeper into the specific statistical learning models used in developing PoS taggers, it may be helpful to the nontechnical reader that the general idea of statistical learning on machines be explained.

Coincidentally, the idea of statistical learning by humans was first formally articulated in the context of linguistics; specifically, language acquisition in infants. Although computational models had been exploiting the implications of this idea in the preceding decade, empirical evidence supporting the concrete existence of such a technique in humans was a result of a study conducted by a group of linguists headed by Jenny Safran in 2003. This study included the experiment of exposing groups of infants less than one year old to speech streams of both “pseudowords” and “nonwords”, each of which contained the same syllables but in different orders. The pseudowords were hypothetical words in the sense that they adhered to the phonotactic constraints of English and would appear to the educated observant as words that were simply made up, while the nonwords could never exist as actual words in English and would appear as foreign gibberish. Upon exposure to these speech streams for only two minutes, the

infants were all recorded as responding differently to pseudowords than they did to nonwords; that is, they were capable of cognitively distinguishing between these two types of words based solely on the order of their syllables after only a relatively small amount of input [Safran 2003]. This implies that the infants were able to, based off hearing carefully crafted examples, learn to make a statistical correlation between the order of certain syllables and the difference between pseudowords and nonwords. While actually documenting and articulating the rules surrounding how the order of these syllables governed a word's likelihood of being a pseudoword or a nonword would be an incredibly difficult task, even for an adult, Safran concluded these rules were nonetheless proved to be easily but intuitively, or subconsciously, learned by mere 8-month-old infants after only two minutes of exposure to speech streams or utterances containing such words.

This idea of circumventing the programming of "hard" or known rules in order to identify or distinguish one thing, or class, from other classes by instead feeding actual examples or parameters, which inherently exemplify the distinctions to be made, to a model inclined to extract the rules or features that define the distinctions, can generalize very well to create statistical systems that do exactly this. While, in the case of the infants in Safran's experiment, the statistical model underlying human cognition is a mystery buried far too deep in human psychology and neuropathology to be well-documented and understood, this does not stop the statistician from using existing mathematical models or creating new ones that are artfully designed to achieve roughly the same, general goal - identifying or distinguishing between different things or classes based on the different

features that the models are able to extract from a variety of different inputs or parameters.

More technically, a statistical learning model may have a variety of parameters, one or more of which serve as references to the raw input, itself, and the others are used to mathematically optimize the model around the input during training. The models discussed in this paper will use supervised training, as opposed to unsupervised training, which roughly translates to the method of, before deploying the model in practice, training the data with input that is said to be labeled. In other words, the statistician or mathematician first trains the model with data whose desired class is known to the model, so that the model can gradually adjust its other parameters in a way that slowly maximizes its predefined accuracy or performance function, or minimizes the error function based on comparing the desired output, or label, to the actual output. It is this optimization during training that allows the model to implicitly, or explicitly in some advanced models, to, analogously to the children in Safran's study, learn the set of features that differentiate some classes from others. The goal of statistical learning is that, after adequate supervised training of the model with labeled input, testing the model with new data will yield a sufficiently low error-rate and high accuracy-rate so that it may then be deployed as a predictive model with practical applications. A simple example would be an Artificial Neural Network (ANN), which will be discussed further in great detail later in the next chapter, being trained to distinguish between pictures of people and animals. The labeled input would be the pixels of images of people and animals, and since the output of most models are numeric and regularized, the images of people will have the label '1' and those of animals will have the label '-1'. As the ANN initially

repeatedly fails to provide the correct output, although in this case it has a 50% chance because there are only two classes, it will gradually optimize its nodes' weights and biases, and perhaps additional parameters if further optimization techniques are used, to minimize the calculated error until the input of a picture of a person has a very high likelihood of receiving the desired output of '1', and animals the desired output of '-1'. It can then be said that the network has effectively extracted the features of what people and animals look like, which are exemplified mathematically in the new structure of the network and the new, corrected values of its weights, biases, and perhaps additional parameters.

This process can of course be calculated by hand. However, in the case of Neural Networks and other algorithms, it can take up to hundreds or even thousands of iterations of training to achieve an acceptable accuracy-rate. Therefore, statistical learning models can instead be expressed with computer programs and algorithms which, being able to perform several computations per second, each of which may take minutes if done by hand, are able to expedite the training and optimization process significantly and save hours, if not days, of time. It is this idea of modeling statistical learning algorithms on computers, in order to not only perform the computations with lightning speed, but also enable the training data to be methodically captured and organized as well as expedite other preprocessing and optimization procedures, that constitute the frequently-cited buzzword and broad concept in computer science referred to as machine learning. Therefore, machine learning is simply a programmatic way to conduct statistical learning on a computer in a way that allows unprecedented avenues for high-speed preprocessing, training, and optimization techniques in a much more manageable and flexible

programmatic environment. It should now be clear that the often cited-together concepts of statistical learning, machine learning, pattern recognition, and artificial intelligence go hand-in-hand. In a nutshell, machine learning allows for the implicit learning of certain distinctions without the features underlying those distinctions being explicitly hard-coded into a computer program. Rather, generalized statistical learning methods are explicitly modeled by computer programs, and may be used in a wide range of applications requiring the program to "learn" the features necessary to make desired identifications and distinctions. One of these applications happens to be learning how to classify given lexical items, or words, into their rightful lexical categories or parts of speech.

Chapter 3: Common Threads in Existing Methodologies

Various machine learning approaches to PoS tagging, including mainly Hidden Markov Models (HMM's), the Viterbi Algorithm, ANN's, and various co-implementations of each will be discussed in this paper. However, it is imperative to first note the common threads which run through all of these algorithms as they are deployed in the PoS tagging context. These themes or procedures include sentence splitting and word tokenization, a finite tagset, as well as the sources of testing and training data used by these various approaches.

There is a crucial first step required between obtaining a text to tag and being able to begin any machine learning-related preprocessing, then the actual training and testing of the machine learning model. This first step is sentence splitting [Tsuruoka]. In her paper, Tsuruoka explains that the cornerstone of a successful PoS tagger, an indispensable prerequisite to further NLP, is the sentence. However, a corpus or body of text is regarded by a computer as simply one contiguous string of text. Therefore, individual sentences must be separated. Sentence boundaries represent the key to the potential for successful PoS tagging, as trying to tag an entire corpus as one continuous sentence would be a syntactic nightmare and would undermine any learning algorithm's ability to extract meaningful features and statistics regarding the likelihood of certain words to fall into one word class rather than another. Additionally, because the algorithm needs to make deductions about words, themselves, isolating sentences is not enough. After isolating sentences via sentence splitting, the words inside of each sentence must be isolated as well, as these words will serve as the input for any given machine learning

algorithm. As for sentence splitting, Tsuruoka recommends the often-used heuristic of indicating sentence boundaries by identifying periods, question marks, and exclamation points followed by one or more spaces and then a capitalized word. Colons and semicolons also must be handled carefully. As for word tokenization, Tsuruoka offers more heuristics such as separating possessive endings and abbreviated forms into their own tokens, and quotes, commas, and other punctuation marks must also be searched for and separated from the word tokens to which they may be attached. Various other methods may be used to effectively tokenize words and interior punctuation.

The next step in creating a PoS tagger is defining the lexical categories one wishes to search for and tag. These tags could be as simple and general as nouns and verbs to more-complex and narrowed tags like past-participle forms of verbs and relative pronouns. Whatever finite set of tags the programmer chooses to deploy in his or her PoS tagger is referred to in research as the *tagset*. As one could imagine, the tagsets of different PoS taggers could be of various sizes; however, this does not seem to be the case with the PoS taggers discussed in this paper, all of which use single or multiple machine learning models. The reason that many machine learning based PoS taggers tag for the same number is because the number of tags must be equal to the number of tags in the training data. Corpora of training data for PoS tagging is very limited, as only a few exist, namely the Brown Corpus and the Penn-Tree-Bank (PTB). However, reducing the number of tags of existing corpora is possible through writing a program that reduces more-specific parts of speech to the more-general categories to which they belong. For this reason, the nearly 78 lexical categories used in the Brown Corpus are usually, but not

always, reduced to the 36 contained in the newer PTB [Derose, 1990; Deisner, 2008; Manning, 2011].

This last point serves as a suitable segue into the final common thread running through most existing and previously constructed PoS taggers, many of which already have and will continue to be referenced throughout this paper. This thread is that the trustworthiness of these corpora, themselves, are of questionable integrity, and that these tagged or annotated corpora whose data is robust enough to facilitate the training of machine learning algorithms is extremely limited. Even the best and most-accurate of PoS taggers in existence, being that they rely on statistical learning models, use either the Brown Corpus or PTB. This is a key point to keep in mind and will be used in conjunction with the previously identified common threads of PoS taggers to form a suitable argument explaining the seemingly asymptotic behavior exhibited by their accuracy rates.

Chapter 4: Various Machine Learning and Mixed Approaches

One of the most predominant machine-learning methods used in PoS tagging is the Hidden Markov Model (HMM). Arguably the single-most notable advantage of the Hidden Markov Model is its ability to handle ambiguous words, or words that, if isolated, may belong to multiple lexical categories or considered to be multiple parts of speech until they are understood in the context of a sentence, and are therefore looked at as an argument of their surrounding words [Tsuruoka]. This is especially important because, as Deisner notes, over 40% of word instances in the Brown Corpus are ambiguous in this sense [Deisner 2008].

However, Deisner is just as convinced as Tsuruoka that using stochastic models like the HMM and implementing it through the Viterbi algorithm is the only way to deal with such ambiguity. In his 2008 paper, he compares the accuracy of two PoS taggers he constructs using a simple HMM in one, and the Viterbi algorithm in the other. He begins by giving the reader a straightforward explanation of the HMM, noting that it exploits the fact that the part of speech of the words in sentences containing ambiguity rely highly on that of one another. The HMM exploits this by traversing the sentence's words, using the current word's predecessors inherent in the training data and acting on a probabilistic function of the model that finds a sequence that maximizes the likelihood of both word probability and tag sequence probability. This initial word probability as explained by Lawrence Saul is calculated as a preprocessing objective using the training corpus, and serves as the model's basis for word probability, also referred to as the Unigram Model [Saul 1997]. While completing a project for AT&T Labs, Saul focuses more on the

underlying mathematics of such variants and cornerstones of the HMM, and describes his construction of a mixed-order Markov Model, which also utilizes Bigrams and Trigrams.

However, Deisner maintains a higher-level description of his project as he continues to explain a bit more in-depth exactly how word probability and tag probability are both maximized using initial probabilities of the word's proceeding that of the current one. He explains that the "hidden" part of the HMM in the context of POS tagging refers to the hidden states of the true tag sequences the model wishes to unveil. This unveiling is referred to as "decoding", which means the following: "given a set of Observations(X) and model (m), we want to reveal the underlying Markov chain that is probabilistically linked to the observed states" [Deisner 2008]. The model consists of three parameters: Initial state probabilities, or a vector to quantify the probability of the first hidden tag in a sentence, and is usually simply the most statistically likely tag if it has no predecessors; state transition probabilities, which are stored in a 2x2 matrix and quantify the likelihood of observing the current hidden state given the previous one; and state emission probabilities, stored in a confusion matrix and specify the probability of observing a particular word while the HMM is in a known hidden state. The training of the model with learning data provides parameter estimation and is a visible Markov process, because the visible words and underlying POS sequences are, at that point, observed. Applying Markov Model training to an unknown sentence to determine an unknown PoS sequence is what makes it a *Hidden* Markov Model. Deisner builds and trains an algorithm based on the Brown Corpus, achieving a 93.5% accuracy but affirming that, with some touch-ups, a rate as high of 97% would be possible. This would be almost as high as the accuracy of the PoS tagger designed by Christopher Manning of Stanford,

which incorporates a similar HMM approach but also uses a plethora of additional post-processing rules to achieve an accuracy rate of between 97% and 98% [Manning 2011]. Manning, however, spends less time describing his model and more time describing the theoretical and practical constraints which account for his model's error. He even goes as far to hypothesize that the accuracy rate of any PoS algorithm is subject to an asymptotic upper bound of slightly below 99%. This claim and the paper's argument in its entirety will be returned to and analyzed more in Chapter 5.

Researcher Jun Wu of John Hopkins University even built a complex, mixed model HMM to be used for PoS tagging, and then used several other taggers to learn the errors of the original model and apply corrections through forming and applying their own rules [Wu 1997]. However, he only managed to decrease the error inherent in the original model by 11%. Although this proved his error-learning models to be somewhat effective, it was a relatively negligible improvement considering the error was already less than 4%.

Deisner [2008], describes yet another and newer statistical approach to Part-of-Speech Tagging, the Viterbi Algorithm:

Viterbi algorithm $\delta_j(t) = \max_X P(X, O, X_t = j \mid \mu)$

where $X = X_1 \ldots X_{t+1}$ and O = output sequence = $O_1 \ldots O_{t+1}$

1. Initialization $\delta_j(1) = \pi_j, 1 \le j \le N$
2. Induction $\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t)\, a_{ij}\, b_{ij o_{t+1}}, 1 \le j \le N$

   Store backtrace $\psi_j(t+1) = \arg\max_{1 \le i \le N} \delta_i\, a_{ij}\, b_{ij o_{t+1}}, 1 \le j \le N$

   where $\psi_j(t)$ = storage of node of incoming arc to most probable path
3. Termination and path (most likely tag sequence) readout (by backtracking)

$$\hat{X}_{T+1} = \arg\max_{1 \le i \le N} \delta_i(T+1)$$

$$\hat{X}_t = \psi_{\hat{X}_{T+1}}(t+1)$$

$$P(\hat{X}) = \max_{1 \le i \le N} \delta_i(T+1)$$

While such statistical approaches are implicitly complicated to the nontechnical reader, the Viterbi Algorithm can thankfully be broken down into plain English. According to Deisner, the Viterbi algorithm is simply a forward algorithm that relies on back-pointers to find the globally best path through what is called the trellis, or a search space in the form of a matrix containing all hidden states and transitions along a sequence of observed states. This algorithm's purpose is to examine every full path through the trellis by recursively finding the partial probabilities from one state to the next, producing the path that is globally optimal. It builds on the Unigram Model's initial state probabilities, which serve to identify the part of speech in the first word of a sentence. It then finds the most probable PoS sequence based on these initial probabilities and the resulting raw, Hidden Markov Model.

Another commonly used statistical learning approach in PoS tagging is the Artificial Neural Network (ANN's). While many find Neural Networks particularly hard to understand, this paper should suffice in giving the reader a general overview of how they function. For this overview we turn to David Skapura's 1995 paper on how to construct such networks and how they function. On a very high level, these networks, which are trained with labeled data in order to build a predictive model meant to eventually classify unlabeled input data similar to that with which it was trained, consist of layers of input nodes, hidden nodes, and output nodes. Leaving out more granular detail like the purpose of activation functions, biases, and hyperparameters aside, the training data travels through the network starting at the input node(s), where it is then used as part of a formula containing weight-matrix and bias values of hidden nodes,

before finally arriving at the output node. This output is then compared to the expected output, and the error is calculated. Backpropagation can then be performed based on the weights of previous node layers, and gradients calculated via partial derivative with respect to each node's respective contribution to the overall error. The weights are then updated by a "learning rate" minus their respective error, so that during the next iteration the node will naturally contribute less to the error. Over many such iterations, the number of which is determined by the learning rate, the error grows smaller and smaller until the weights are optimized for the classification of the data being trained - resulting, hopefully, in a reliable, predictive model that can be used for detecting or classifying information similar to that with which it was previously trained.

Like HMM's, ANN's are a classical and well-documented approach to PoS tagging. One of the first neural networks to be introduced into this context [Schmid 1994] opened the floodgates for what would produce ample research on the use of ANN's in performing this task. In addition, ANN's have not just been used for English part of speech tagging, but also in Portuguese [Marques 1995]. However, these taggers also exhibited similar accuracy rates which, along with even the most effective of previously discussed HMM's and even mixed approaches, all converged, at their highest, around 97%.

Chapter 5: Critique of Former Methodologies: The Need for Linguistics

To this day, based on what has been published and is unclassified, no researchers in academia or the private sector have been able to construct a PoS tagger that defeats this upper bound on accuracy of approximately 98%. Christopher Manning [2011] of Stanford University explicitly addresses this problem, and pinpoints the reasons for inaccuracy. He argues and provides ample evidence from research that 81% of the error that accounts for the ~2% inaccuracy of even the best of PoS taggers can be broken down into five categories: "Could plausibly get right" (16%), "Difficult Linguistics" (19.5%), "Underspecified/Unclear" (12%), "Inconsistent/no standard" (28%), and that the "Gold standard is wrong" (15.5%) [Manning 2011]. This chapter will break down and analyze each of these categories using Manning's guidance as well as intuition concerning linguistics and the machine learning tactics that result in these errors, both respectfully and jointly, with the goal of seeking a different PoS tagging methodology that would reduce, or perhaps eradicate altogether, these errors.

The first error mentioned was referred to by Manning referenced instances that the algorithms "could plausibly get right", yet did not. There is not much to be said about this error, except that it is undoubtedly caused by and/or related to the two errors he mentions next: "Difficult Linguistics" and, "Unspecified/Unclear". Together, these three errors constitute 37.5%, or nearly 40%, of the remaining gap between the hypothesized ~98% accuracy upper-bound and the desired 100% accuracy rate. The reasons behind this large portion of the error in question, if resolved, would be a large enough improvement to disprove the hypothesis of a 98% asymptote. These reasons revolve around the

machine learning approach to a purely linguistic problem. First of all, as previously cited in this paper, the size of the tagsets used for machine learning algorithms in the realm of PoS tagging must match that of the available training data, the least of which being the PTB corpus with a tagset size of 36. Having such a large tagset makes the potential for accurate tagging exponentially harder. As discussed in Chapter 6, the concept of word classes as described by prominent Linguistics, if they concede that the abstraction of word classes exist at all, should be viewed as a continuum. A fine analogy for this dilemma is the spectrum of color. Imagine if one were to color a picture with seven crayons, each of which represent a basic color of the pallet of the rainbow. If this sheet of paper were to be then passed to even a child, "tagging" each color with its correct label would be quite trivial. However, imagine if you used a box of 128 crayons. Even an adult with the sharpest eye in the world would have trouble correctly labeling each of the 128 colors. This is because the distinctions made between color, or cuts in the continuum, are too close together. The same can be said for the abstraction of parts of speech. The more tags that an algorithm tags for, the more room for error. Another reason that machine learning is at the roots of these errors is because it is statistically based, and therefore is heavily biased against exceptions or *outliers*, which are quite common when one is dealing with language. If the algorithm conjures from training data that word *x* is a noun 98% of the time, word *x* will certainly almost always be tagged as a noun. But what about that other 2% of the time? These exceptions will be neglected, resulting in errors.

After describing these first three errors, the fourth error which Manning identifies is "Inconsistent/No standard". This is a particularly interesting type of error whose origins are an unarguable result of using statistical learning as the primary approach to

PoS tagging. Assume again that there exists a word *x*, except this word is not contained in any of the training data fed as initial input to the statistical learning algorithm. When it comes time for prediction, and the algorithm encounters word *x*, it has no chance of being correctly tagged unless its part of speech happens to match the default in the post-processing section of the algorithm used for words that cannot be resolved. For an example of the potentially devastating implications of this inevitable manifestation, the Viterbi algorithm implemented by Deisner [2008], when encountering words never before seen in the training data, would lead to zero probabilities for entire vertical columns in the trellis, causing crucial probability paths to break and accounting for an entire 4% decrease in accuracy. Because there is no way that the limited amount of annotated corpora available for training machine learning algorithms for PoS tagging can possibly encompass every lexeme that the resulting predictive model may encounter in the wild during its application, it is no surprise that this error, alone, accounts for 28%, or more than a quarter, of the error bridging the gap between 98% and 100%.

Perhaps the most troubling error that Manning mentions is the fifth and last one, which asserts that "the Gold Standard is wrong." By the "Gold Standard", Manning is referring to the available annotated corpora like the Brown Corpus and the PTB, which, to reiterate, are a couple of the only corpora available that can be effectively used as training data for a statistics-based or machine learning model. When he asserts that these corpora are "wrong", he is revealing the disturbing discovery from his research that a fair bit of the words contained in the corpus are incorrectly tagged. This means that, in addition to the numerous other errors that plague PoS tagging algorithms relying on statistical learning models, some of the training data used to obtain initial probabilities

and draw other conclusions is simply incorrect. Manning traced the impact of this incorrect or disputed data used in training his models to result in 15.5% of the total error exhibited by his algorithm. He concludes that this is simply unacceptable.

So what can be learned from Manning's observations, and how can any drawn conclusions be used to formulate a better approach to PoS tagging? This chapter has highlighted two crucial themes that will be used to develop a new approach, which can be embodied by a single statement: Machine learning as an approach to PoS tagging, due to the nature of its supervised training and the current state of available and applicable training data in the form of annotated corpora, is what is leading to the currently cited "upper bound on accuracy" inherent in such algorithms. To elaborate, the success of statistical learning based approaches to solving problems are subject to the training data that shape them. In the case of PoS tagging, there are several problems with the training data, namely the Brown Corpus and PTB, that this chapter has shown to be empirically linked to the approximately 2% error in even the most sophisticated of PoS tagging algorithms which use this data. These problems are the unnecessarily high amount of tags, the inaccuracies inherent in the training data, and the inability of pure statistics to deal with complex linguistics which may contain statistical outliers. Due to these problems, which are specific to machine learning models because of their dependence on the limited and flawed amount of available training data, it would be logical to conclude that the notorious upper-bound on accuracy in PoS tagging only exists so long as this training data is used in the training of predictive models.

Taking this conclusion and its empirically-based evidence into consideration, it becomes obvious that the upper-bound on accuracy of PoS tagging algorithms can

theoretically be beaten with a surprisingly simple-sounding, but actually very difficult-to-implement approach. The technical details of this approach are reached by simply examining the problems elucidated in this chapter, and deploying certain heuristics and methodologies to circumvent them. An algorithm of this kind would have to abandon machine learning, altogether, due to its dependence on the fundamentally flawed data by which it would be forced to train. Once the algorithm is free from the constraints previously imposed upon it by the corpora typically used to train it, the programmer is free to determine his or her own tagset. In other words, this tagset could be much smaller. In addition, free of statistical obligations, the algorithm could deploy complex rules along with exceptions in order to handle more-difficult linguistic problems and complex sentences. Exactly how such an algorithm could be written will be explained in-depth in the coming chapters, along with additional theoretical arguments to affirm its features.

Part Two:
A New, Linguistic and Rule-Based Approach

Chapter 6: Strategically Defining a Minimal Tagset

The purpose of this paper is to create, or at least lay down the foundation for, a theoretical approach to PoS tagging that is not subject to the asymptotic upper bound of PoS tagging accuracy expressed by prominent researchers like Wu [1997], Deisner [2008], and, namely, the developer of the most-accurate PoS tagger known to date, the Stanford Part of Speech Tagger - Dr. Christopher Manning [2011]. As touched on at the end of Part I of this paper, the key to unlocking such an accurate PoS tagger would be one built from a purely linguistic approach with ambiguity resolved by rules, whether simple or complex in their nature. This solely rule-based approach to PoS tagging has been hypothesized as literally "impossible" by many prominent researchers, including Manning and the previously-cited Tsuruoka. In conducting thorough research for this paper, not one instance of a purely rule-based approach was found. These facts imply that creating such an algorithm or system would be truly venturing into unchartered territory. For this reason, much of the remainder of this paper will lack sources; however, it will utilize the sources at hand to coalesce some of their various techniques in the realms of sentence splitting and word tokenization, as well as draw from some of the already touched-upon linguistic theory concerning word classes. By combining proven algorithmic techniques and incorporating linguistic knowledge as the primary, not secondary, mode for successfully tagging lexemes with their correct parts of speech, a

PoS tagger that more resembles human intuition and reflects deep knowledge of the English language is hoped be achieved.

As explained in Chapter 1, one of the first things a programmer wishing to build a PoS tagger must do is define his or her tagset. As a result, as explained in Chapter 3, this is one of the few common threads running throughout all PoS tagging algorithms. While the tagset is seldom looked-upon as a feature crucial to accuracy, this paper contends that the quantity of the tags within the tagset is an obvious parameter to accuracy. But as previously pointed out, due to the nature of the statistical methods analyzed in Chapter 4 and scrutinized in Chapter 5, the quantity of the tagsets of PoS taggers utilizing these algorithms is fixed. This fixed quantity is determined by the size of the tagset exhibited by the training data used to train the algorithm in the case of ANN's, or set the initial word-and-PoS relation probabilities and frequencies in the case of HMM's. Because annotated training data for PoS algorithms is limited to the few corpora including mainly the Brown Corpus and the PTB, the quantity of these algorithm's tagsets range from numbers in the 30's to the 70's. However, to the trained and educated Linguist, distinguishing word classes from one another to this degree is not only trivial and impractical, but contrary to the very nature of word classes. This idea goes as far back to the writings of Linguists like J.D. McCawley in the mid-to-late 20th century, to the most-recent writings of modern, prominent Grammarians like David Denison [McCawley 1982; Denison 2014].

Denison may be the leading authority on the subject of why parts of speech are not the easy-to-define, definite aspects of a word which Computer Scientists would like them to be. In his 2014 paper titled "Parts of speech: Solid Citizens or slippery

customers?", Denison conducts a deep dive into the generalization we call "word classes", or parts of speech, which people have theoretically constructed in an attempt to narrow their understanding of language. This is not to say that he believes the distinction is useless, as he concedes that parts of speech are extremely useful for people studying and using a language, especially if it is not their native language. Still, he urges linguists to view these distinctions for what he believes they truly are - not isolated spheres or groups of words, but rather as a continuum characterized by the underlying and ever-changing semantics and syntax of the given language. In this rationale, Denison makes the observation that some words lying between the most-basic word classes should not be technically labeled as a single class, but rather as a ratio of the classes it may exhibit in certain usages. He explains that "PoS represent abstract generalisations about the behavior of . . . thousands and thousands of words", and that "there cannot be a rigid distinction between open classes like Noun, Adj, V, which accept new items freely" [Denison 2012]. This claim that language naturally resists being neatly classified into even the most simple word classes has been backed historically by many Linguists with an expertise in this field, as McCawley [1982] claims that parts of speech or as he refers to them, "syntactic categories", do not even exist at all. Other linguists like Paul Hopper [1997], have made great strides and have written and spoken at-length simply to describe and prove the existence of a single word class. In 1997, Hopper wrote a journal entry titled "Discourse and the category Verb in English", in which he makes a hard-to-debate argument that a Verb is, in fact, a part of speech in English due to its never-failing tie to what Hopper refers to as an "Event". He argues that without a Verb being in a sentence,

the sentence cannot effectively describe anything actually taking place, thus defining the word class in a new light.

Therefore, Hopper and many other Linguists aren't quite as pessimistic about word classes as people like McCawley, including Denison. Denison concludes his 2012 paper by admitting the key to distinguishing between word classes, and the very few word classes he believes to be well-enough defined to draw distinctions between, as he only refers to as Nouns, Adjectives, and Verbs, is syntax. In other words, how words are syntactically distributed in relation to one another and within the context of the sentence is the only way to truly distinguish one word class from another. *Phrases* are the basis of syntax and the study of their various structures serve as a cornerstone in the study of syntax. For example, the noun phrase (NP) structure is discussed in-depth by Payne [2002]. His primary concern, however, is the crux of ambiguity in the English language - zero derivation, or conversion. This is the phenomenon of one word class gaining new, already-existing recruits. To the non-Linguist, a simple example would be the company Google, whose search engine has grown so popular in this era that people frequently use the company's name as a verb meaning to search on the internet, or, *to Google.* This phenomenon of zero derivation, which will from here forward be referred to as "conversion", will be handled methodically by this paper's prescribed PoS tagger and will be discussed later. For now, the quantity of the tagset is to be the focal point of attention.

It should be clear now that the dozens of tags used by previously-analyzed PoS taggers undoubtedly contributes greatly to their total errors. Even when two humans attempt to manually tag the Brown Corpus with all of its tags, for example, the total

disagreement rate between them is about 7.2%, almost twice the error of even mediocre machine learning algorithms trained on the same corpus [Manning 2011]. This demonstrates why too many parts of speech could be a problem. Two linguistics could argue for hours over whether a certain word is a transitive adjective or a preposition, but certainly even two children would have no disagreement about whether the word *hit* in the sentence "*He hit me"* is simply a verb. Additionally, less parts of speech are actually, and this may be a shock to many Computer Science PhD's with only secondary knowledge of Linguistics, of more practical use to Linguists - especially in such fields as discourse analysis. When analyzing large corpora in order to find patterns, or validate or dispute already-hypothesized notions, the typical Linguist does not need more than a dozen, and probably not even a dozen, word class options to choose from and correlate. These linguistic analysts are primarily interested in connecting certain agents to their actions (or vice versa), or adjectives to whom or that which they describe. This really requires only a few parts of speech with some integrated NLP, at the very most. While a Computer Scientist building an AI engine with an advanced NLP engine may require more parts of speech at his or her disposal, this is not necessarily a tool which can be used by actual Linguists who may not have much technical knowledge with regards to NLP. They may be interested in AI, but the basic technology they need in order to do everything from recording the exact pitch and air waves of phonemes in the field, to analyzing phonology and other areas of Linguistics in a lab or even conveniently in their offices, has existed for years. What Linguistics do not have, however, is a 100% accurate, or even 99% accurate, PoS tagger that they can trust to tag their custom, self-aggregated corpora so that they may conduct advanced analysis on it. Nevertheless, this is not to say

that a 99%-100% accuracy PoS tagger, even with less tags, would not be beneficial to some areas of NLP. It is simply a matter of making a trade-off between being satisfied with a pretty accurate PoS tagger that can tag dozens of word classes but has a definite upper bound of around 98%, or choosing only a dozen or so of the word classes it absolutely *needs* to eliminate the asymptotic bound which is currently said to cap the accuracy with which sentences and corpora can be tagged.

With that said, Dennison [2012] was not speaking about PoS tagging when he claimed that analyzing syntax was key to determining a word's part of speech, so his liberalism regarding the continuity of word classes and the only three specific ones he mentions must be taken with a grain of salt by the programmer wishing to use his advice. To elaborate, using syntax, or the word's surrounding syntactic arguments in a sentence marked for only Nouns, Verbs, and Adjectives to deduce this unknown word's part of speech would be extremely difficult at best, and borderline impossible at worst. Therefore, it is imperative for the programmer designing the PoS tagger to walk the thin line between too few and too many word classes when constructing his or her tagset, making only enough distinctions so that a suitable amount of rules can be created, and any remaining "unknowns" after initial processing have the highest likelihood of being tagged accurately.

Walking this line, the author of this paper has chosen to loosely follow the guidelines for the eight parts of speech described by Jeff Glauner [2002] in terms of both the parts of speech form and function. Glauner also explains the morphological processes we sometimes use to break down morphemes, as does Deisner [2008] as a post-

processing step in his algorithm. This point will be returned to and utilized later in the paper.

Glauner's eight parts of speech become nine parts of speech when subclassing the Definite Determiner class, containing *the, a,* and *an.* This will make it easier to identify Nouns as these Determiners always precede and refer to a noun, while the placement of other adjectives in the sentence, even ones that may function as Determiners sometimes, can be syntactically ambiguous from a PoS tagging point of view. This number then grows to ten when we include Ordinal Numbers, or mixtures of letters and numbers, such as *2nd, 2016,* etc. - distinguished again from the rest of the Adjectives. This number again grows to eleven when we distinguish the interrogative WH- pronouns and *how* from the rest of the Pronoun class. Finally, a word class for Negation, or negative markers (e.g. *not*) and modals (e.g., *might, could* etc.) are added because they are comprised of short lists of known elements and can help to better deduce the PoS of their surrounding arguments. The thirteen parts of speech which will constitute this paper's PoS Tagger's tagset are defined below by their traditional one-liner definitions, and their forms - which will be used to construct useful rules for tagging them within a sentence [Glauner 2002]:

- **Nouns [N.]**
    - Traditional Definition "A word that names a person, place, or thing."
    - Form:
        - may end with two inflections - plural and genitive
        - may also end in nominal derivational suffixes

- **Pronouns [PR.]** (Including **Interrogative Pronouns [IP.]**)
    - Traditional Definition: “A word that takes place of a noun.”
    - Form:
        - May be personal, demonstrative, indefinite, -WH pronouns, *how*
- **Verbs [V.]** (Including **Modal Verbs [MV.]**)
    - Traditional Definition: “A word that shows an action or state of being”
    - Form:
        - may take verbal inflections
        - may have derivational affixes

- **Adjectives [ADJ.]** (Including **Determiners [Det.]** and Ordinal Numbers **[OD.]**)
    - Traditional Definition: “a word that modifies a noun or pronoun”
    - Form
        - may take comparative or superlative suffixes, or preceded by *more* or *most*
        - may end in derivational adjectival suffix
- **Adverbs [ADV.] (**Including **Negative Marker [NEG.]**)
    - Traditional Definition: “A word that modifies a verb, adjective, or another adverb.”
    - Form:
        - may be intensifiers or degree adverbs

        - may take comparative endings; can be preceded by more/most
        - positions are very flexible within the sentence
- **Prepositions [Prep.]**
    - Traditional Definition: “The first word of a prepositional phrase”
        - Finite, closed class of words
- **Conjunctions [Con.]**
    - Traditional Definition: “join words, phrases, or clauses”
    - Form:
        - Contain coordinating conjunction, correlative conjunctions
- **Interjections [Int!]**
    - Traditional Definition: “express emotional sentiment of speaker”
    - Form
        - Often short outbursts, not grammatically structured as to be a stand-alone sentence
        - Often end in an exclamation mark

This completes the tagset to be used for this paper’s rule-based PoS tagger, and their respective definitions and abbreviations, which will be used for the output of the actual program. The algorithm will take a sentence as input, and output a sequence of the above abbreviations, which may be used going forward in this paper to refer to their respective parts of speech. To reiterate, the thirteen parts of speech for which the PoS Tagger will be tagging are: Nouns (N.), Pronouns (PR.), Interrogative Pronouns (IP.),

Verbs (V.), Modal Verbs (MV.), Adjectives (ADJ.), Determiners (DET.), Ordinal Number (OD.), Adverbs (ADV.), Negative Markers (NEG.), Prepositions (PREP), Conjunctions (CON.), and Interjections (INT!). The significance of this tagset, to reiterate on a point made frequently in this paper, is its relatively small quantity - while walking adequately the line between minimal quantity and useful distinctions. While this tagset's quantity is just over one third of the size of the next-smallest tagset that has come up in the research conducted to write this paper, it does not make the tagset so small that important distinctions are blurred and the case for practical use is marginalized. Although it does generalize many of the granular parts of speech cited in other papers higher-up into the PoS hierarchy, the concept of 'word class" as viewed like a continuum is still intact, despite the smaller amount of discrete points should it be graphed in such a way. At the same time, it does leave some seemingly less practical parts of speech and granular distinctions like determiner and modal verb intact, the reason behind doing so is that linguistic rules can be more-easily defined within the PoS tagging algorithm in order to resolve the parts of speech of words in the input sentence whose word classes remain unresolved and unknown even after initial testing and PoS recognition tactics. Therefore, the size of and distinctions within this tagset are relatively optimal to obtaining the two, often conflicting chief goals it must accomplish - making enough granular distinctions to create meaningful linguistic-based rules used for the actual PoS tagging, while simultaneously remaining small and generalized enough to be of practical use to the actual Linguist or client once the PoS tagging has successfully taken place. This point of view reveals the angle that the size or enlargement of the tagset is not only inversely proportional to the accuracy of the PoS tagger, but that it has a similar relationship with

how practically the tagger can be put to use. Therefore, by maintaining a relatively small tagset but still making just a few granular distinctions to improve accuracy by building intricate rules where necessary, this PoS tagger is able to walk the tight-rope between practicality and theory without letting either side dominate and inhibit its potential for realistic use.

Chapter 7: Establishing Methodology

Every PoS tagging algorithm, whether stochastic or rule-based, must have a well-defined methodology for fulfilling its purpose. The methodology used to tag the words of a sentence with their parts of speech for this paper's algorithm can be broken down into a four-step, sequential process. These steps are sentence parsing and string tokenization, lexeme or whole-word comparison of each tokenized word with a pre-installed lexicon of words with known word classes, bound-morpheme or derivational affix comparison of the beginning or end of each remaining-to-be-tagged tokenized word with a pre-installed lexicon of derivational affixes which are known to always transform the stems to which they are attached into known word classes, and, finally, the application of syntactical and grammar-rooted rules in an attempt to deduce the word class of any remaining tokens whose parts of speech is still unknown.

For the first step of tokenizing the input sentence into words and punctuation, the sentence boundaries, themselves, are first identified by a sentence-terminating punctuation mark (i.e. a period, question mark, or hyphen), followed by one or more spaces, and finally a capital word. This ending punctuation mark itself is saved as a token, which terminates the sentence, because its nature will become critical later when defining rules for the final step. After a sentence has been extracted, it is tokenized by spaces to come up with individual words. However, what this paper refers to as *interior punctuation*, such as commas, semicolons, colons, quotation marks, hyphens, and may also still be in the sentence; but, thankfully, these are all attached to words and can be

easily identified. These punctuation marks are tokenized as tokens themselves, with two exceptions: Hyphens are ignored because they are meant to join multiple lexemes into a single lexical item of its own. The second exception is the apostrophe, in which case the word containing the apostrophe will be checked against the short list of all-known contractions, then separated into its two constituent words, each of which being correctly marked for their part-of-speech. For example, if *shouldn't* is in the sentence, *should* and *n't* are separated into their own tokens and tagged as [MV.] and [NEG.], respectively.

The second step calls for plain String comparison of each word in the input sentence to pre-processed, imported lists of words with only one, known part of speech. This is equivalent to giving the algorithm a kind of lexicon containing lexemes whose word classes are never or rarely ambiguous. This works especially well for lexical categories including, but not limited to, closed word classes like Prepositions and Pronouns, whose constituents are finite, unambiguous, and can be listed briefly. Additionally, this tactic can still be used on more-open, larger word classes like Nouns and Verbs, with the lists ordered decreasingly by frequency and capped at an appropriately large number with the hopes of at least identifying a couple of them within the input sentence before resorting to more-complex methods. Yet, one must watch out most closely here for "the crux of ambiguity" in English, described by Dennison [2012] as conversion between Nouns and Verbs. Therefore, any Nouns or Verbs included in the lexicon for comparison must be unambiguous, or unable to be used as either word class based on the context of a sentence. This can be achieved by implementing a pre-processing method which removes lexemes contained in both the Noun and Verb lexicons.

The third step in the process is almost identical to the last step with respect to its String comparison to an already constructed and simply-imported lexicon. However, that which is being compared this time are the derivational affixes - or the prefixes and suffixes, if any, attached to each word in the input sentence, with another lexicon filled with hundreds of these affixes and their denoted parts of speech which each affix forces the stem to which it is attached to transform into. For example, any word ending with the derivational affix -*tion* is a noun. By using lists of derivational affixes that force their words into specific word classes, scanning the makeup of each word in the input sentence to check for these affixes becomes incredibly useful in PoS tagging. Another very common example is the suffix -*ly*, which signals the word is an adverb with a few exceptions such as *friendly*, which can always be hardcoded as an exception to this general rule.

So what happens after the aforementioned three steps are applied perfectly, yet there are still words whose word class is unknown to the algorithm? This is when the last and most-important step comes in. Instead of using any statistical or machine learning processes, the algorithm described in this paper will attempt to deduce the word class of the final "unknowns" using purely linguistic logic and rules drawn from the studies of grammar and syntax, as well as the intuition of Linguists. These rules are built around attempting to tag the unknown word class of word *x* in the input sentence by using the resolved parts of speech of the lexemes, and sometimes the lexemes themselves, surrounding *x* as parameters to the rule. The nature, structure, and organization of these rules vary significantly, and the list can grow to be very long and complex. Therefore, it would be inappropriate to simply list the rules here. Instead, here is a simple example

written in plain English, and not yet hardcoded as a rule to be applied: *If there exists a determiner, then there must exist a noun somewhere after it in the sentence*. This draws upon the Linguistic knowledge that determiners are reserved for referencing the head of the next-closest noun phrase in any sentence in which a determiner appears. By using simple to very complex rules which draw exclusively from Linguistic knowledge, this paper contends it is possible to resolve all remaining "unknowns" after first applying the first three steps of its prescribed methodology, and may very well circumvent the often-cited asymptotic upper-bound of accuracy for PoS tagging incurred by Computer Scientists and Mathematicians who rely heavily on statistical models and flawed corpora and incorporate very little, if any, linguistic knowledge into their approaches.

Now, the generalized four-step sequence which this paper's algorithm will use to approach the problem of PoS tagging should now be understood as relatively straightforward. The first step is to actually parse the sentences and tokenize their words and interior punctuation. The next two steps involve traversing the tokenized words as well as their affixes and comparing them with parts of an artificial lexicon whose lexemes and derivational affixes are associated with a known word class, and assigning this word class to the word under analysis if there are any matches. Finally, to resolve any remaining unknown words whose morphology did not match that of anything in the lexicons to which they were compared, the algorithm will use a series of Linguistic rules based on the grammar and syntax of the English language in an attempt to logically deduce the word classes of any remaining words. The next chapter will be devoted to exactly how this system will be designed and implemented in a way that enables these four steps to take place in a simple, but algorithmically efficient fashion.

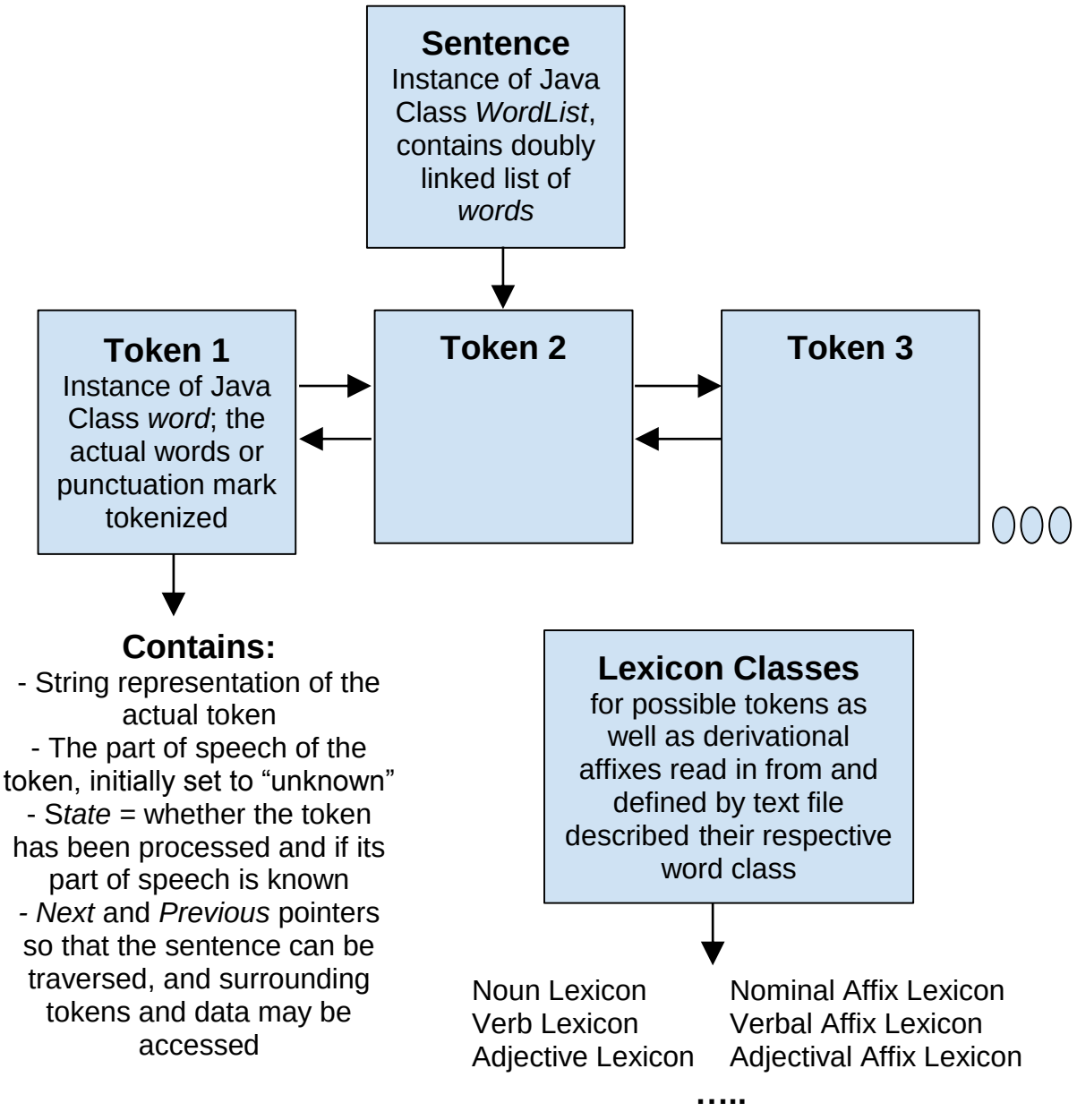
Chapter 8: Design Specifications and Implementation

Before diving into technical detail about a system's design, it is sometimes helpful to present a diagram that roughly models the system to reveal the intuition belying its structure [Figure 1].

As seen in the figure, the key class concepts that are custom-created for this algorithm are *WordList* and *Word.* These two constructs are designed with the previously discussed methodology in mind, and lend themselves nicely to the process described in the last chapter. While a WordList object is the collection which is meant to parse the tokens of the input sentence into a doubly-linked list, the Word object is the token itself, along with other data members about the token and its maturity in the algorithm's progression, which are crucial to monitoring and carrying out the prescribed steps of the PoS tagging process.

By storing each parsed sentence into a *WordList* of *Words,* the program is able to examine each word according to its own properties and related to its morphology and state in the progress of the PoS tagging algorithm, as well take into account its syntactic position relative to other words and their properties by using iteration and utilizing the words' *previous* and *next* pointers during the rule application phase. In addition, multiple *Lexicon* classes extended from an abstract class allow text files of known words, affixes, and their parts of speech to be converted to a HashMap for constant-time access during the comparison phase of the algorithm.

Figure 1


Sentence
Instance of Java Class *WordList*, contains doubly linked list of *words*
Token 1
Instance of Java Class *word*; the actual words or punctuation mark tokenized
Token 2
Token 3
Contains:
- String representation of the actual token
- The part of speech of the token, initially set to "unknown"
- *State* = whether the token has been processed and if its part of speech is known
- *Next* and *Previous* pointers so that the sentence can be traversed, and surrounding tokens and data may be accessed
Lexicon Classes
for possible tokens as well as derivational affixes read in from and defined by text file described their respective word class
Noun Lexicon
Verb Lexicon
Adjective Lexicon
Nominal Affix Lexicon
Verbal Affix Lexicon
Adjectival Affix Lexicon
.....

Perhaps the single most important data member in regulating the algorithm is the *state* of each word, initially set to "unchecked" upon instantiation. As iteration throughout the sentence occurs, the word's state is changed to either "checked" or "known", depending on whether the word has been checked, but not resolved, as a syntactic argument of its surrounding words, which can be based on either their resolved parts of speech or, in some, cases, the actual lexemes. An entire explanation of the variety, complexity, and extent of these rules would warrant another theoretical, indefinitely lengthy discussion outside the scope of this paper. Nonetheless, the algorithm, after checking the words against its lexicon during the first iteration, proceeds to apply a plethora of linguistic, grammatical rules to the words whose states are checked but still unresolved. Contrary to what one's intuition may conceive, the formulation and application of such rules, as discussed later in this paper, have never before been documented or attempted by other researchers due to their reliance on statistical modeling in PoS tagging. A notable exception is Christopher Manning's addition of rules on top of Stanford's primarily statistical approach to its PoS tagging algorithm [Manning 2011], but these rules are not elaborated upon, much less revealed. With that said, it is important to note that Manning admits that it is rules like these that must be the key to breaking the asymptotic nature of so many PoS tagging algorithms, which rely entirely on machine learning methods and statistical models. Here is an example in order to illustrate the kinds of rules the author(s) of this paper have logically conceived, and how they are programmatically applied.

Logical Representation of Rule:
$\forall$.word $\in$ DET. $\Rightarrow$ $\exists$.X $\in$ N. s.t. X follows word

English Representation Rule:
For each and every word whose PoS is a Determiner, there must exist a corresponding noun which it references later in the sentence.

Potential Application Rule:
If there exists a Determiner and an "unknown" word later in the sentence, and there are no other nouns following this determiner, then the "unknown" word must be a Noun.

Another simple example, taken from Payne [2010], is that if a word is preceded by the lexeme *very*, there is nearly an over a 99% probability that the part of speech of the word in question is an adjective - as *very* is an intensifier. It could also be used to modify an adverb as opposed to an adjective, but adverbs can be ruled out first quite easily by examining their ending for the typical *-ly* derivational suffix.

For simplicity, there will be no attempt to apply rules on a Boolean basis and reconcile discrepancies. Instead, rules will be implemented in nested if-else statements inside a while-loop controlling the constant iteration and application of rules to the sentence in an attempt to resolve all parts of speech. Because there is no feasible way to prioritize and order all of the rules to be applied in a way that holds true for all sentence structures, it may take several iterations and applying of all rules before all of the expected pieces fall into place, enabling other rules to play their part. For this reason, the termination condition for the external while-loop and continuation of the last phase of the algorithm on an input sentence is not only (1): That the state of all words is "known", but also (2): That all rules have been attempted to be applied at a liberally-estimated twenty times. If the algorithm terminates on condition (1), then it has reached what it thinks is a

solution to the PoS problem, and will print each original word followed immediately by a forward slash and its predicted PoS evaluation, with each of these tokens separated by a space. If the algorithm terminates on condition (2), there will be word(s) whose part(s) of speech are unknown, and an appropriate error message will be displayed to aid subsequent programmers in determining the reason for the error, so that they may pursue further addition, deletion, or tweaking of certain, relevant rules within the algorithm. This constant tweaking of this proposed algorithm to improve its accuracy in paramount to eventually achieving a fully-functioning, reliable PoS tagging algorithm over time. This idea is precisely the theme of the next and final chapter.

Chapter 9: Measuring Accuracy and Success

At this point, the reader should rightfully expect the accuracy rate of the algorithm that has been described and set forth throughout Part II of this paper, and has been implemented accordingly by the author, to be stated as a plain statistic. However, the purpose of this dissertation has been to argue against the hypothesized asymptotic upper-bound on accuracy of PoS tagging algorithms, by examining the common threads and reasons for errors which have permeated the statistics-based PoS tagging algorithms that have exhibited this aforementioned upper bound on accuracy, and exploiting this research to propose a prototypical algorithm that would deviate from these commonalities and therefore not exhibit this upper-bound. With that said, this proposed algorithm is built upon a basis which is purely theoretical at this point, with the current version of the algorithm not easily lending itself to being, at least for now, tested to produce a pinpointed accuracy rate, for a variety of reasons. This chapter will discuss the two prevailing reasons why, although the algorithm has been implemented, it cannot and should not be measured for accuracy at this time, but must rather be continue to be implemented over time in order to confirm the theoretical argument that its eventual accuracy can conquer the upper bound proposed and exhibited by the work of multiple researchers cited in this paper, including Wu [1997], Deisner [2008], and Manning [2011]. The bases of these aforementioned two reasons have already been discussed in this paper, so their implications for the immediate testing of the algorithm described in Part II of this paper should not be surprising.

Perhaps most important, the first reason that the proposed, implemented algorithm cannot be currently tested is related to the nature of the available corpora that have been pointed to numerous times already and even described as the Gold Standard by researchers such as Manning. As already discussed, mostly in Chapter 5 and Chapter 6, the annotated corpora against which PoS tagging algorithms may be checked pose numerous problems. The first problem is that the size of the tagsets of these corpora and that of the algorithm proposed in this paper do not match. That is, due to the method of increasing accuracy by decreasing the size of the tagset, the number of tags in the corpora available for testing accuracy are at least three times the number of tags for which this paper's algorithm tags. That is, if the linguistic, rule-based algorithm described in the previous chapters were to attempt to tag such corpora, there would be varying inconsistencies leading to errors, manifested solely because the tags inherent in the algorithm proposed in this paper are simplified or reduced versions of the more-granular tags annotated throughout existing corpora. For example, consider the word *ran* in the simple sentence "I ran". If this sentence exists in the Brown Corpus, The word *ran* would be tagged as a past-tense verb. However, the algorithm set forth in this paper does not make such granular distinctions. Instead, it would simply tag the word *ran* as a verb. Of course, both of these tags are correct; however, in a test for accuracy, this discrepancy would result in an error which would artificially lower the accuracy rate of the latter algorithm. Therefore, all of these instances, when compounded, would cause significant damage to the algorithm's overall accuracy rate. This artificial decrease in the accuracy of the algorithm would also be exacerbated by the previously-discussed fact that corpora like the Brown Corpus are riddled with errors, or incorrect annotations.

The solution to this problem would be to check the algorithm against a corpora annotated with *only* the tags of which this algorithm's tagsets consists. Unfortunately, no such corpora currently exist. As equally-as-unfortunate, the manual composition of such an annotated corpus would have to be annotated by-hand and possess a word-count in the thousands, if not tens-of-thousands, in order to produce a statistically significant accuracy rate for algorithms checked against it. Such a task would not only be incredibly tedious, but would also take such a long time that doing so would violate the author's academic timeline governing the completion of this thesis and a functional implementation of the algorithm described therein.

The second reason why the algorithm should not be currently tested for accuracy as means for confirming or refuting the argument in this paper lies in the nature of the algorithm itself. The success of this algorithm in accurately tagging parts of speech relies heavily on the linguistic rules which are programmed into its logic. In the relatively young study of Linguistics, there has not been much attention paid to the fact that the abstraction of word class, or part-of-speech in English is deeply rooted in complex syntax and semantics of the language. The methodology of constructing rules involving components such as syntax in order to systematically resolve the lexical category of words, in context, would require a new, revolutionary approach to examining grammar and the rhetorical structure of sentences in the English language [Firth 1968]. While Firth's paper is almost ancient when compared to the timeline of the study of Linguistics, Terence Parsons [1990] of MIT would later resurface the idea of the nature of lexical categories in his study of "subatomic semantics", described in a paper in which he, although successfully linking a word's part of speech in any given sentence with its

subtle semantic nuances, and demonstrating that semantic change always precedes and implies the potential of a word to flourish in an entirely new word class, fails to give any insight into how these connections could be used for identification of a word's part of speech [Parsons 1990]. In other words, attempting to construct hard-coded rules and successfully identifying their exceptions, if any, in order to successfully implement an algorithm meant for PoS tagging is unchartered territory in the field of Linguistics. This is perhaps due to the fact that these algorithms must run on a computer, forcing the problem to be forwarded to Computer Scientists - few of whom are prepared to delve into unchartered linguistic territory and instead resort to statistical methods with maybe a few linguistic heuristics which are rudimentary, at best.

Therefore, the crux of creating the kind of linguistic, rule-based PoS tagging algorithm described in this paper is forming novel and truly new rules, often involving complex analysis of English grammar, based primarily on intuition due to the lack of relevant research on this topic. This reliance upon intuition implies that the algorithm should not be a static entity, but rather a constantly evolving solution that is frequently updated with new rules as linguists receive new insight into the systemic nature of parts of speech. This process could take years or even decades, as more and more attention is given to the problem and research collaboration between linguists is directed towards finding its solution. However, regardless of the timeline, this paper and the first prototype of its described algorithm should suffice in serving as the theoretical argument and foundation for breaking the fallacious upper-bound on accuracy described by many scientists who have failed to develop PoS tagging algorithms which fall short of 100% accuracy.

With that said, the abandonment of the statistical models, "gold standards", and excessively granular tagsets used by these researchers and a return to pure linguistics in solving what is an unarguable linguistic problem may take time, but is theoretically the only way to finally achieve the 100% accurate PoS tagging algorithm which researchers have been scrambling to create for decades. It is evident that this path to success is a long one, and requires considerable advancement in human understanding of how grammar is methodically structured according to not only phrases', but individual words' parts of speech. However, if this path is taken, and its algorithm is constantly approved upon as primarily linguists, not computer scientists, spend a considerable amount of their time and devote much of their attention to solving this problem, and what could potentially exist as an entirely new field in Linguistics, achieving 100% accuracy in the tagging of words of their part of speech is theoretically possible. Not only would this require and lead to unprecedented advancement in collective humanity's understanding of language, but would also propel the algorithm's applications in Natural Language Processing, Human-Computer Interaction, and other revolutionary fields to new, and much-needed heights.

Bibliography

Christopher D. Manning. 2011. Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? *Computational Linguistics and Intelligent Text Processing Lecture Notes in Computer Science* (2011), 171–189.

David Denison. 2014. Parts of Speech: Solid Citizens or Slippery Customers? *Online Journal of the British Academy British Academy Lectures* 2012-13: 133-68.

David M. Skapura. 1995. Building Neural Networks. ACM Press/Addison-Wesley Publ. Co., New York, NY, USA.

Helmut Schmid. 1994. Part-of-speech tagging with neural networks. *Proceedings of the 15th conference on Computational linguistics.*

Jana Deisner. 2008. Part of Speech Tagging for English Text Data. Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Jeff Glauner. 2002. *Essentials of Grammar: A Textbook for Teachers, Editors, Secretaries, Writers, and Other Semiwilling Curmudgeons.* Second ed. 2002

Jun Wu. 1997. Using Multiple Taggers to Improve English Part-of-Speech Tagging by Error Learning. John Hopkins University, Baltimore, MD, USA.

J. Payne. 2002. Nouns and noun phrases. *The Cambridge grammar of the English language* Cambridge University Press, Cambridge. 323–523.

J. Payne. 2010, The distribution and category status of adjectives and adverbs. *Word Structure,* 3. 31–81. DOI: http://dx.doi.org/10.3366/E1750124510000486

J.D McCawley. 1982. The nonexistence of syntactic categories, Thirty million theories of grammar. London, Croom Helm. 176–203.

J.R. Firth 1952-59. 1968. A new approach to grammar. I*n F. R. Palmer, ed., Selected Papers of J.R. Firth, 1952-59,* (1968). Longman

Lawrence Saul. 1997. Aggregate and mixed-order Markov models for statistical language processing. AT&T Labs - Research, Florham Park, NJ, USA

Nuno M.C. Marques. 1995. Part-of-speech tagging for Portuguese texts. Advances in Artificial Intelligence Lecture Notes in Computer Science (1995), 323–332.

Paul Hopper. 1997. Discourse and the category Verb in English. *Language and Communication. Special Issue: The Importance of Theory in Discourse Analysis. Vol. 17,2*: (1997), 93-102.

Terence Parsons. 1990. Events in the Semantics of English: A study of Subatomic Semantics. The MIT Press.

Steven DeRose. 1990. Stochastic Methods for Resolution of Grammatical Category Ambiguity in Inflected and Uninflected Languages. Brown University Department of Cognitive and Linguistic Sciences, Providence, RI

Yoshimasa Tsuruoka. Sentence splitting, tokenization, language modeling, and part ... Retrieved February 5, 2016 from http://nactem.ac.uk/dtc/dtc-tsuruoka.pdf