MJUR

Unsupervised Part-of-Speech Tagging: An Introduction

Alex Brooks
Monmouth College

Mike Stees
Monmouth College

The process of assigning parts of speech to words based on context is known as part-of-speech tagging. One traditional approach is to use labeled instances of text in order to train an algorithm for the part-of-speech tagging process. However, labeled data is often very difficult or time consuming to obtain depending on the domain. That said, unlabeled data can be relatively easy to collect. Unsupervised learning uses strictly unlabeled data to solve a problem. In this paper we provide an introduction to statistical unsupervised part-of-speech tagging with a focus on methods using Hidden Markov Models and extensions proposed in some of the relevant literature to date. We provide some historical background on corpora development and automated tagging, discuss the use of lexicons in unsupervised methods, present a general method using Hidden Markov Models, and conclude with a presentation of extensions and other approaches.

1. Introduction to Part-of-Speech Tagging

As early as elementary school, humans learn about various parts of speech and how those parts of speech help define the structure and meaning of a sentence. Oftentimes, many words have many different associated parts of speech, each of which can be expressed with labels, or *tags*. The word *model* can be described as either a verb or a noun, for instance. In order to determine which tag is associated with *model*, a context must be given. In the sentence *The class was going to model the experiment based on previous research*, *model* is a verb. However, in the sentence *The model of the boat was almost complete*, *model* is a noun. The process of assigning parts of speech to words based on context is known as *part-of-speech tagging* [4]. Part-of-speech (POS) tagging is one of the early tasks in processing natural languages. Some of these tasks include parsing, information extraction, and information retrieval. Small improvements in the tagging process have the potential to yield larger improvements overall in many advanced natural language tasks, since POS tagging is a vital pre-processing step [10]. For this reason, research has focused on practi-

cal methods that yield the highest accuracies for POS tagging.

This paper provides an introduction to statistical unsupervised part-of-speech tagging with a focus on methods using Hidden Markov Models and extensions proposed in some of the relevant literature to date. The paper is organized as follows. Section 2 will provide some history on automated tagging and corpora[1] development, specifically the Penn Treebank and the Brown Corpus. Section 3 will introduce unsupervised learning, discuss the use of lexicons in unsupervised POS tagging, and present a general method for unsupervised POS tagging. Section 4 will review some extensions of the method presented in Section 3, as well as other statistical approaches that do not use HMMs. Finally, Section 5 will present a non-statistical approach.

2. History of Corpora and Automated Tagging

Part-of-speech (POS) tagging has been performed by humans for decades. Most research in POS tagging resulted from work on corpora or large sets of structured text. One key use of corpora in natural language processing tasks requires sections of text to be annotated. Annotated corpora are usually known as Treebanks or Parsed Corpora. The most popular annotated corpora today are the Brown Corpus and the Penn Treebank.

The Brown Corpus was first prepared between 1963 and 1964 and is currently available in six versions, each containing the same basic text consisting of over 1 million words. Form A, the original version, simply contained text which was not tagged. Form B, the second version, is a stripped version of Form A, where all forms of punctuation symbols, excluding the following, have been removed: hyphens, apostrophes, and symbols for formulas and ellipses. Form C, the third version, is the first to be tagged and have an associated tagset[2] which is found in Appendix A. The final three versions, Bergen Form I, Bergen Form II, and Brown MARC Form, are all versions which contain minor revisions of the first three versions. Overall, the corpus is divided into 500 samples, each consisting of 2,000+ words. Every sample begins at the beginning of a sentence but not necessarily at the beginning of a paragraph or larger contextual divider. Once a sample exceeds 2,000 words the sample is ended after the final sentence, is complete. Some of the texts included are press reports and editorials, religious texts, general fiction texts, science fiction texts, and romance and love story texts. [6]

The Penn Treebank project was developed between 1989 and 1996 at the University of Pennsylvania. The corpus includes approximately 7 million words of tagged text, 3 million words of skeletally-parsed text, over 2 million words of text parsed for predicate-argument structure, and 1.6 million words of transcribed speech annotated for disfluencies. The treebank uses a tagset of 48, found in Appendix B, where 36 tags are allocated for parts of speech and 12 tags are allocated for punctuation and currency. Some of the annotated texts included in the treebank are Department of Energy scientific abstracts, Department of Agriculture bulletins, IBM computer manuals, and a retagging of the Brown Corpus material. [16] [26]

Although both corpora provide the same functionality, there are many differences between the Brown Corpus and the Penn Treebank. For instance, the Brown Corpus allows for compound tagging and the Penn Treebank does not. Consider tagging the contraction *I'm*. By using the Brown Corpus tagset, *I'm* would be annotated with two tags, *I'm*/PPSS+BEM [6], whereas using the Penn Treebank tagset, *I'm* would be split and separately tagged, *I*/PPSS *'m*/BEM [16]. Although the Penn Treebank tagset is based off of the Brown Corpus tagset, many tags and properties of the set were removed [16]. For instance, compound tagging was removed in favor of splitting contractions into the corresponding separate parts, as seen above. Further, the Brown Corpus tagset includes tags that are specific to particular lexical items. For example, the Brown Corpus distinguishes three forms of *do* and eight forms of *be* [16]. Although the goal of the Brown Corpus was to be robust in distinguishing tags for certain lexical items, the goal of the Penn Treebank developers was to remove possible redundancies from the Brown Corpus tagset. As such the developers attempted to make the Penn Treebank more concise, resulting in a reduced tagset of 48 tags [16] [26].

Considering the size of the Brown Corpus and the Penn Treebank, tagging either corpus is by no means a simple feat. To elaborate, consider the sentence

The lazy gray sloth climbed up the tree (1)

There are two main ways to visualize the tagged version of (1): by an in-text tagged sentence or by a parse tree. Using the Penn Treebank tagset, we can generate the corresponding tagged sentence and parse tree for (1), as seen in (2) and in Figure 1, respectively.

The/DT *lazy*/JJ *gray*/JJ *sloth*/NN *climbed*/VBD *up*/IN *the*/D T *tree*/NN (2)

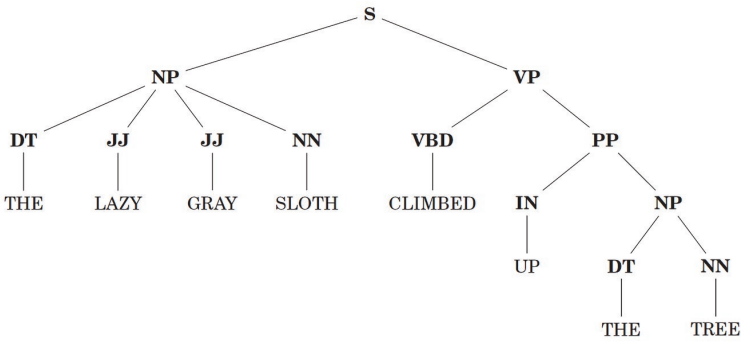

Figure 1
Parse Tree for *The lazy gray sloth climbed up the tree*

Regardless of which approach is considered, there are clear complexities in tagging even this simple sentence. Consider the possibility of having three distinct tags for each word in (1). By disregarding context, this results in 6,561 possible tag sequences for this single sentence. For an average sized paragraph, this escalates to 3×10^{30} possible sequences. With this in mind, there is an inherent difficulty in hand tagging large sets of data. Consequently, interest arose in automating the tagging process by using computers beginning in the 1950s and 1960s [5]. One of the earliest programs for performing automatic POS tagging was the TAGGIT system designed by Greene and Rubin in 1971 using a transformation-based[3] approach [4]. The system was a major step in fully tagging the Brown Corpus generating Form C. The program was initially run on the corpus, correctly annotating roughly three-fourths of the text, and the remaining text was manually annotated [4]. As research continued, entirely automatic transformation-based methods began to arise. Most systems required the use of *lexicons*, or dictionaries, which contained every possible tag for the words that were provided [5]. The systems annotated text by first assigning every possible part of speech to all of the words in the text. Then, the sequence of rules was consulted, eliminating tags which were not possible in the given contexts. Multiple passes through the text were done using the rules until each word was annotated with only one tag.

There are many issues associated with hard-coding rules for transformation-based methods. They do not transfer nicely to new tagsets, languages, or corpora [5]. This is because not all tagsets have the same tags and not all languages have the same words or words with the same meaning. Further, sentence structure for languages can be different and thus the context for rules do not transfer easily. Finally, the transformation-based methods require that humans write the rules, which can be a very lengthy and complicated task. For these reasons most of the research transitioned away from transformation-based methods and focused on statistical approaches. Although these statistical approaches will be the focus of this paper, there will be some discussion on the transformation-based methods in Section 5.

3. Unsupervised Part-of-Speech Tagging

Unsupervised learning methods learn patterns with the absence of explicit feedback [22]. Unsupervised methods operate on data that have not been tagged with expected results. When applying unsupervised learning to POS tagging, the only information available in some cases is a lexicon. In other cases no lexicon is provided and the process relies completely on the given data.

In the following subsection a discussion on the use of lexicons in unsupervised learning methods is provided. Afterward, a general unsupervised learning method using Hidden Markov Models (HMMs) for POS tagging is presented. For readers unfamiliar with HMMs, please refer to Appendix C.

3.1 Lexicons in Unsupervised Learning

It has been shown in [1] that using a reduced lexicon greatly improves the tagging accuracy of unsupervised methods. This is because filtering the lexicon removes tags that are statistically unlikely to occur. As a result, noise is removed from the lexicon and processing becomes more efficient and more accurate, since the more statistically likely cases are favored. However, as pointed out by [1], obtaining a reduced lexicon is dependent on some form of human intervention. In other words, the method can no longer be classified as unsupervised as it requires some form of manually tagged data or a lexicon that has been manually edited by a human. Future research might consider focusing on methods of coping with noisy lexicons.

The use of lexicons in general should be further discussed within the context of unsupervised methods. An important thought to consider is how the use of any kind of lexicon affects the classification of a POS tagging method. Reviewing our definition of unsupervised learning, feedback is not received on any decisions made by the tagger. Although using lexicons does not result in any direct feedback on any attempted tagging, it certainly limits the possible classifications a word can be given. With this in mind, any results presented in the following sections will be clearly separated based on whether the lexicon is a full lexicon or an optimized lexicon.

3.2 Generalized Process with Hidden Markov Models

In part-of-speech tagging, we can use HMMs to model the dependency of the context of words based on associated lexical tags [4]. This is done by estimating the model parameters A (state transition probabilities), B (emission/observation probabilities), and Π (initial state probabilities) from a training set. The state transition probabilities represent the probabilities of a certain tag following a given tag and the emission probabilities represent the probabilities of a word being labeled a tag. The "hidden" aspect of the HMM refers to the set of states. The "observed" aspects of the HMM are the sets of words for each tag in a lexicon. Since, in any given context, only one part of speech is correct for a single word, we must use the HMM to identify the most likely sequence of tags associated with our set of text.

In the example in Appendix C, the employer set the model parameters. This is not done for unsupervised part-of-speech tagging. Rather, we must begin with a set of, usually uniform, probabilities and train the HMM to reflect statistically accurate sets of probabilities. Through training the HMM we maximize the probabilities of the parameters, a process known as *maximum-likelihood estimation* (MLE). The maximization is done by using an algorithm known as the Baum-Welch or Forward-Backward algorithm [4]. This algorithm iteratively constructs a sequence of models (sets of parameters) which improve the probability of the training data [18]. This algorithm is labeled as an *Expectation Maximization* method for MLE.

Let N be the number of possible tags, o be the output sequence (the sentence), and o_t represent the individual words in the sentence at "time" t. The algorithm begins by recursively defining two sets of probabilities: the forward probabilities:

$$\alpha_j(t+1) = \sum_{i=1}^{N} \alpha_j(t)\, a_{ij}\, b_{j(o_t+1)} \tag{3}$$

Where $b_{j(o_t)}$ is the emission probability given state j at time t and backward probabilities:

$$\beta_j(t) = \sum_{j=1}^{N} \beta_j(t+1)\, a_{ij}\, b_{j(o_{t+1})} \tag{4}$$

where $b_{j(o_{t+1})}$ is the emission probability given state j at time t+1 and $\beta_j(T) = 1$ for all j, where T is the final time in the iteration. At time t, the forward probability $\alpha_i(t)$ is the joint probability of the sequence of observations $\{o_1, o_2, \ldots, o_t\}$ up to time t and the likelihood of observing o_i at time t. Similarly, the backward probability $\beta_j(t)$ is the probability of the sequence of observations $\{o_{t+1}, o_{t+2}, \ldots, o_T\}$ given that we observe o_j at time t. It follows that the probability of the entire sequence $\{o_1, o_2, \ldots, o_T\}$ is

$$P = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_j(t)\, \beta_j(t+1)\, a_{ij}\, b_{j(o_{t+1})} \tag{5}$$

Note that (5) holds true regardless of the value t. Further, given an initial choice for A, B, and Π, we may compute the probability of a transition between state s_i at time t and state s_j at time t+1, identified as $\gamma_{ij}(t)$:

$$\gamma_{ij}(t) = \frac{\alpha_i(t)\, a_{ij}\, b_{j(o_{t+1})}\, \beta_j(t+1)}{P} \tag{6}$$

By summing γ_{ij} for all t=1...T, we have the expected number of transitions from state s_i to state s_j in the sequence. This allows us to estimate the values for a_{ij}, b_{jk}, and π_i:

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_{ij}(t)}{\sum_{t=1}^{T-1} \sum_{m=1}^{N} \gamma_{im}(t)} \tag{7}$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \hat{b}_{jk}(t)}{\sum_{t=1}^{T-1} \alpha_i(t)\,\beta_j(t)} \quad \text{where} \quad \hat{b}_{jk} = \begin{cases} \alpha_i(t)\,\beta_j(t) & \text{if } k = o_t \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\pi_i = \frac{\alpha_i(1)\,\beta_j(1)}{P} \tag{9}$$

To clarify, equations (7 - 9) represent the following information [21]: is

$$\alpha_{ij}: \frac{\text{the expected number of transitions from state } s_i \text{ to } s_j}{\text{the expected state transitions from } s_i}$$

$$\beta_{jk}: \frac{\text{the expected number of times in state } s_i \text{ and observing } k\ (o_t)}{\text{the expected number of times in state } s_i}$$

π_i : the expected frequency in state s_i at time $t = 1$

In most cases the initial probabilities for the parameters are uniform [28]. That is, the state transition parameters are uniform over all tags and the emission probabilities are uniform over the set of possible words for a given tag.

In summary, to find the ML estimates for our parameters A, B, and Π using the Baum-Welch algorithm, we choose some starting values and then apply equations (7 - 9) to compute new values. We iterate through this process until the parameters converge to maximum values. Note that regardless of the initial parameter values, the parameters will always converge to a local maximum [28]. Once the parameters are maximized we can produce the most likely sequence of tags for our text. This process can be thought of as maximizing over all sequences which might generate a sequence of tags S [4]. The algorithm we will be using to perform this task is a dynamic programming algorithm called the Viterbi algorithm [27].

In order to find the most probable sequence of states V for the given sequence of observations O we must define the following quantity:

$$\phi_i = \pi_i\, b_{i(o_1)} \text{ for } i \le i \le N \tag{10}$$

$$\phi_i(t) = \max_{v_j \in V, j=1,\ldots,(t-1)} P(v_1 \ldots v_t = i,\ o_1 \ldots o_t) \tag{11}$$

This represents the highest probability along a single path at time t, ending at state s_I and accounting for the first t observations. By induction we can show

$$\phi_i(t+1) = \max_i (\phi_i(t)\, a_{ij}) \cdot b_{j(o_{t+1})} \tag{12}$$

To retrieve the state sequence, we need to keep track of the argument which maximized equation (12) for each t and j. To do so, we must define a new function, ψ

$$\psi_i(1) = 0 \tag{13}$$

$$\psi_j(t) = \underset{1 \le i \le N}{\operatorname{argmax}} (\phi_i(t-1)\, a_{ij}) \quad \text{for } 2 \le t \le T,\ 1 \le j \le N \tag{14}$$

Finally, to find the most probable sequence of states or *Viterbi path* we must perform backtracking,

$$v_T^* = \underset{1 \le i \le N}{\operatorname{argmax}} (\phi_i(T)) \tag{15}$$

$$v_T^* = \psi_{v_{t+1}^*}(t+1) \quad \text{for } t = T-1,\ T-2,\ \ldots,\ 1 \tag{16}$$

For more information on the algorithms and other applications of HMMs see [21].

4. Extensions and Other Statistical Models

We have divided the extensions of approaches using HMMs into Expectation Maximization (EM) and Bayesian. Following the extensions are other statistical approaches to unsupervised POS tagging, which do not require a HMM, including clustering, prototyping, and methods for multi-lingual POS tagging.

4.1 Expectation Maximization

For most problems using HMMs, there are many mappings between an underlying distribution and the distribution regarding the observation [19]. The EM algorithm seeks to produce Maximum Likelihood Estimates (MLEs), or estimations of the parameters of a statistical model. The EM algorithm is defined in two steps: an expectation step followed by a maximization step [19]. This first step is with respect to the unknown variables and uses the current estimates of the parameters [19]. The second step provides a new estimate [19]. The algorithm is iterative and thus steps 1 and 2 are repeated until the estimates converge.

The first modifications we will present come from [15] and are extended in [13] [14]. The goal in [15] was to introduce *equivalence classes* to reduce the number of parameters in the model, alleviating problems in obtaining reliable estimates for individual words. Each class contains a set of words that have the same parts of speech. For example, words that can only be nouns belong to the *noun class*, words that can be nouns or verbs belong to the *noun-or-verb class*, and so on. The number of equivalence classes required becomes independent of the size of the dictionary, accommodating new words without any need for retraining [14]. By using equivalence classes, dependency on the subject domain is removed as well. Rather than assigning probabilities to single parts of speech and training the HMM accordingly, parts of speech are replaced with the equivalence classes. Each equivalence class is initially assigned equal probabilities. Once the HMM is trained, the probability for an equivalence class is divided evenly between the parts of speech the class represents. For example, if the equivalence class *noun-or-verb* class has a probability of 0.3, the noun and verb parts of speech each receive 0.15 probability. By using equivalence classes within the HMM, dictionaries containing hundreds of thousands of words can be represented by only a few hundred classes. The results in [14] report errors as low as 4% for an optimized lexicon and errors around 22.9% for a full lexicon [1].

Another modification to the typical HMM is presented in [1]. The goal of [1] was to introduce more context into tagging. Originally, the probability of a word was conditioned only on the current tag. However, there may exist dependencies between a word and the parts-of-speech of the words preceding and following it. With this in mind, [1] modified the structure of the HMM to estimate the probability of a word based on the tags immediately preceding and following it. They called this structure a *contextualized* HMM. By increasing the context size during the estimation, it was found desirable to further smooth the estimations using a technique found in [20]. These modifications result in errors around 6% for an optimized lexicon and 23.1% for a full lexicon [1].

The final two modifications to the EM approach we will present are discussed in [28]. The goal of [28] was to continue using a simple HMM with improvements on the state transition probabilities and emission probabilities, while still using a full lexicon. The first improvement looked at constraining the HMM, specifically the state transition probabilities, to maintain a specified marginal distribution. The goal of this improvement was to improve the quality of the transition probabilities by keeping the parameters to reasonable values [28]. The target distribution is found using the probability of each tag in the complete text being tagged [28]. After finding the target distribution, the M-step of the EM task is modified, adding a constraint requiring that the transition probabilities stay within the defined distribution. By including this constraint, the tagging error rate from a uniform initialization of the parameters improves from 18.7% to 9.5% after several iterations of the Baum-Welch algorithm in each case [28]. These results, however, require having a reasonable tag distri-

bution.

The second improvement from [28] focuses on improving the emission probabilities. The issue found with using the standard EM process is that it does not incorporate any form of "parameter tying" over the word emission probabilities [28]. In other words, the parameters are treated independently, when many words play similar syntactic roles. They add feature vectors for each word, each consisting of mutual information between the word and a context. Once the feature vectors have been determined, the similarity between two words is computed. Then, a similarity measure is used to smooth the parameters by taking similarity weighted averages. Specifically, the emission probabilities are smoothed. By performing this modification, the tagging error rate improves from 18.7% to 12.6%. After combining the two modifications the tagging error is slightly worse than strictly using the first modification, with an error rate of 9.97%.

4.2 Bayesian Inference

Unlike the EM approaches where the parameters are maximized, a Bayesian approach integrates over all possible parameter values [8]. This difference allows the learned structure to have a high probability over a range of possible parameters and permits the use of a prior probability distribution, hereafter termed a prior. Most words are associated with very few parts of speech as previously mentioned, which results in sparse emission probabilities. Priors allow us to preference the models that have these sparse emission probabilities [7]. The goal is to use a prior and the likelihood estimation (Equation (5)) to estimate the posterior distribution. The posterior distribution equates to the probability of the estimated model given a corpus to tag. In most cases, the posterior distribution does not have a closed form. However, there are two main techniques to approximate the distribution: Variational Bayes and Markov Chain Monte Carlo [11]. In [8], Gibbs sampling, a Markov Chain Monte Carlo method, is used to produce samples from the posterior distribution. This process is done by iteratively re-sampling each tag according to its conditional distribution given the current values of all other tags [8]. In contrast, Variational Bayesian inference attempts to find a function which minimizes an upper bound called the Variational Free Energy [11]. We can use the factorized form of the function to accurately approximate the posterior distribution [11]. The process requires only minor modifications of the M-step in the Baum-Welch algorithm.

4.3 Other Models

4.3.1 Clustering

The approach taken in [23] is similar to the contextualized HMM found in [1] where context of a word is increased by adding dependence on the immediately previous and following tags. The context is incorporated through feature vectors labeled left context and right context vectors, respectively [23]. The

difference between [23] and [1] is that the underlying model was not a HMM. The vectors are composed into a matrix and singular value decomposition (SVD) is applied, which is a method for factorizing matrices. For more information on SVD, refer to [12]. There are four different tagging approaches [23] considered, including: induction based on word type only; induction based on word type and context; induction based on word type and context, restricted to "natural" contexts; and induction based on word type and context, using generalized left and right context vectors [23]. For more information on the different approaches see [23].

In context distribution clustering, as presented in [3], each word defines a probability distribution over all contexts, notably the probability of the context given the word [3]. Since data is generally too sparse to estimate context distributions for most words in a corpus, [3] approximates the context distributions as probability distributions over ordered pairs of clusters. In other words, the same idea of introducing context through adding dependencies on the immediately previous and following words is done. In [3], an iterative algorithm is presented to estimate the context distributions, further using ideas from EM and Bayesian Inference to define the categories of a text.

4.3.2 Prototyping

In a prototyping approach, prototypical examples are specified for each target label or label configuration. For example, when modeling the Penn Treebank, the treebank tagset and a few examples of each tag are listed [9]. The general approach taken in [9] is to use distributional similarity to link a given word to similar prototypes. These links are then encoded as features in a log-linear generative model [9]. The model used is a chain of Markov random fields, which are the undirected equivalent of HMMs [9]. For more information on prototype-driven learning applied to POS tagging, see [9].

4.3.3 Multilingual Learning

The key hypothesis of multilingual learning is that by combining cues from multiple languages, the structure of each becomes more apparent [24]. The work in [24] is based on the idea that patterns of ambiguity inherent in POS tags differ across languages. At the word level, a word with tag ambiguity in one language may correspond to a word with an unambiguous tag in another language [24]. Although the learning style can address ambiguities in each language, it must be flexible enough to accommodate cross-lingual variations such as the tagset and the syntax of a sentence [24].

Within multilingual learning, training occurs on parallel data and distinct tagsets. In other words, the training data contains the same text presented in multiple languages where each sentence contains the same content in its respective language. Further, each tagset available to the learner is tied to a specific language. There is a possibility for the tagsets to be the same, however this does not always occur since some languages do not have certain parts-of-speech. For example, Serbian is a language without articles [24].

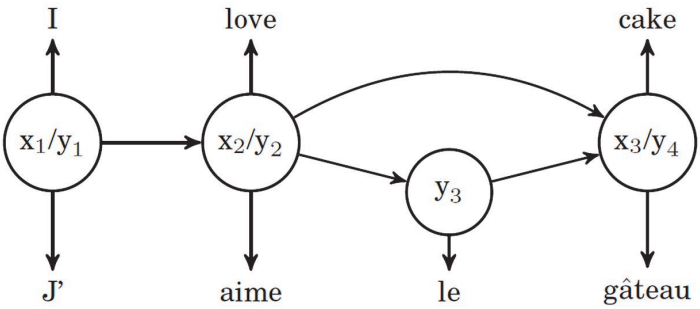

Figure 2
Graphical structure of the bilingual model based on word alignments [24].

A bilingual model for jointly tagging parallel streams of text in two languages is proposed in [24]. The model is designed to permit information to flow across the language barrier [24]. They assume that for pairs of words that share similar semantic or syntactic functions, tags will be statistically correlated [24]. After using techniques from machine translation to align similar words as seen in Figure 2, a hierarchical Bayesian model is formulated in [24] that exploits both language-specific and cross-lingual patterns to explain the observed bilingual sentences. After constructing the model, Bayesian inference is used across the defined parameters to construct the emission probabilities for generating sequences of tags [24]. Tests were run across all pairs of the following four languages: English, Serbian, Bulgarian, and Slovene. Of the tests run, the highest accuracies achieved using full lexicons for each language are 92.01%, 91.75%, 94.48%, and 95.10%, respectively [24]. Tests were performed using reduced lexicons containing only the 100 most frequently used words for all pairs of the four languages, resulting in 71.34% 56.91%, 62.55%, and 59.68%, respectively [24]. All results were improvements on mono-lingual models of the same structure [24].

A multilingual model allowing the inclusion of more than two languages in the training process is proposed in [25]. Three tests were performed on sets of eight languages including Bulgarian, Czech, English, Estonian, Hungarian, Romanian, Slovene, and Serbian, where each test varies the amount of information provided in the lexicon [25]. Results for a full lexicon are less than ideal, improving accuracy of only four out of the eight languages, compared to the bilingual model from [24]. For the two reduced lexicon tests, where words were included based on frequency of the words (frequency of greater than 5 and 10 respectively), five of the eight languages improved in accuracy over the bilingual model [25]. Of all tests performed, English did not improve over the bilingual model.

5. Transformation-Based Learning

Training a part-of-speech tagger using transformation-based learning requires a simple set of deterministic rules [2], as opposed to statistical approaches, which require large sets of probabilities. An unsupervised transformation-based learner is comprised of three components: the initial state annotator, the set of transformation templates, and the scoring criterion [2]. Initially, the learner has access to an unannotated corpus and a lexicon with words and the allowable tags for each word [2]. The initial state annotator tags each word in the corpus with a list of allowable tags. The transformation templates are for reducing the uncertainty of the correct tag of a word in a particular context [2]. All learned transformations in [2] will have the following form:

Change the tag of a word from X to Y in context C (17)

X is defined as a set of two or more tags and Y is a single tag where $Y \in X$. Some example transformations which were learned by the learner in [2] are:

Change tag from NN_VB_VBP *to* VBP *if the previous tag is* NNS
Change tag from NN_VB *to* VB *if the previous tag is* MD
Change tag from JJ_NNP *to* JJ *if the previous tag is* NNS

In the scoring criterion, for each learning iteration, the score of a transformation is computed based on the current tagging of the training set. To score the transformation in (17), for each tag $Z \in X$ where $Z \neq Y$, define the following:

$$F = \frac{\text{freq}(Y)}{\text{freq}(z)} * \text{incontext}(Z, C) \tag{18}$$

Where freq(Y) is the number of occurrences of words tagged with Y in the corpus, freq(Z) is the number of occurrences of words tagged with Z in the corpus, and incontext(Z, C) is the number of times a word tagged with Z occurs in context C in the training corpus [2]. In (18), freq(Y) ÷ freq(Z) represents the relative frequency of tag Y to tag Z. Note that when evaluating (18) for different tags Z, freq(Y) remains constant. Thus for larger values of F, the less likely Y is a better choice than Z (i.e. this occurs when Z is used relatively infrequently for different words but occurs many times in the context of C in the training corpus). Therefore, F gives us a way to quantify the benefit of choosing tag Y over tag Z or vice versa. Now, this calculation is used to find the "best" relative alternative tag choose for the transformation in (17) by defining the following:

$$R = \text{argmax}_Z(F) \tag{19}$$

Finally, R is used to determine the validity of choosing Y. This is done by scoring the transformation as follows:

$$S = \text{incontext}(Y, C) - \frac{\text{freq}(Y)}{\text{freq}(Z)} * \text{incontext}(R, C) \quad (20)$$

This recalculates F for choice $Z = R$ and compares it to the number of times Y shows up in context C.

If (20) is positive, than Y is a better choice than R; the higher the score is, the better the choice Y is for the transformation in (17).

Now that a method is defined for scoring the transformations, the learner calculates the scores for each transformation and chooses the transformation with the maximum score for each iteration. Once no positive scoring transformations can be found, learning is halted [2].

Training on the Penn Treebank and the Brown Corpus, the tagging accuracy converged to an error of 5% after learning over 1,150 transformations, with an initial tagging accuracy of a 9.3% error [2]. While these results are promising, the vast majority of research on transformation-based learning was conducted in the 1990s and interest has since declined.

6. Conclusion

Techniques exploring unsupervised part-of-speech tagging are motivated by the increasing availability of unlabeled data. With current research presenting extensions of a general Expected Maximization method and other approaches for unsupervised POS tagging, accuracies are showing overall improvements. As we discussed earlier, small improvements in the tagging process have the potential to yield larger improvements overall in many advanced natural language tasks, since POS tagging is a vital pre-processing step [10]. Although many different approaches have been developed for the tagging process, unsupervised methods which utilize Hidden Markov Models seem to be the most popular and perform the best. It was our goal in writing this paper to introduce material associated with part-of-speech tagging. In particular by focusing on corpus history, automated tagging, lexicons, a general model, and extensions on the general model for unsupervised POS tagging, we hoped to present a comprehensive introduction to this material.

Notes
1.) Plural of corpus, or large set of structured text
2.) Set of tags
3.) Methods using rules
4.) Hyphenated before regular tag
5.) Hyphenated after regular tag

References

1.) Michele Banko and Robert C. Moore. Part of speech tagging in context. In Proceedings of the 20th international conference on Computational Linguistics, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

2.) Eric Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In Proceedings of the third workshop on very large corpora, volume 30, pages 1–13. Somerset, New Jersey: Association for Computational Linguistics, 1995.

3.) Alexander Clark. Inducing syntactic categories by context distribution clustering. In Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7, pages 91–94. Association for Computational Linguistics, 2000.

4.) Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In Proceedings of the third conference on Applied natural language processing, ANLC '92, pages 133–140, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

5.) R. Dale, H.L. Moisl, and H.L. Somers. Handbook of Natural Language Processing, volume 1. Marcel Dekker, 2000.

6.) W Nelson Francis and Henry Kucera. Brown corpus manual. Letters to the Editor, 5 (2):7, 1979.

7.) Jianfeng Gao and Mark Johnson. A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 344–352. Association for Computational Linguistics, 2008.

8.) Sharon Goldwater and Tom Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In Annual Meeting-Association for Computational Linguistics, volume 45, page 744, 2007.

9.) Aria Haghighi and Dan Klein. Prototype-driven learning for sequence models. In Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pages 320– 327. Association for Computational Linguistics, 2006

10.) Association for Computational Linguistics, 2006.

11.) Nitin Indurkhya and Fred J Damerau. Handbook of Natural Language Processing, volume 2. Chapman & Hall/CRC, 2010.

12.) Mark Johnson. Why doesn't em find good hmm pos-taggers. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 296–305, 2007.

13.) Virginia Klema and Alan Laub. The singular value decomposition: Its computation

and some applications. Automatic Control, IEEE Transactions on, 25(2):164–176, 1980.

14.) Julian Kupiec. Augmenting a hidden markov model for phrase-dependent word tagging. In Proceedings of the workshop on Speech and Natural Language, pages 92–98. Association for Computational Linguistics, 1989.

15.) Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. Computer Speech & Language, 6(3):225–242, 1992.

16.) Julien Kupiec. Probabilistic models of short and long distance word dependencies in running text. In Proceedings of the workshop on Speech and Natural Language, pages 290–295. Association for Computational Linguistics, 1989.

17.) Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: the penn treebank. Comput. Linguist., 19(2):313 –330, June 1993.

18.) S. Marsland. Machine learning: an algorithmic perspective. Chapman & Hall/CRC, 2009.

19.) Bernard Merialdo. Tagging english text with a probabilistic model. Comput. Linguist., 20(2):155–171, June 1994.

20.) Todd K Moon. The expectation-maximization algorithm. Signal processing magazine, IEEE, 13(6):47–60, 1996.

21.) Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modelling.

22.) Computer Speech and Language, 8(1):1–38, 1994.

23.) Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286, 1989.

24.) S.J. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2010.

25.) Hinrich Schütze. Distributional part-of-speech tagging. In Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics, pages 141–148. Morgan Kaufmann Publishers Inc., 1995.

26.) Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Unsupervised multilingual learning for pos tagging. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1041–1050. Association for Computational Linguistics, 2008.

27.) Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Adding more languages improves unsupervised multilingual part-of-speech tagging: a bayesian non-parametric approach. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Associa-

tion for Computational Linguistics, pages 83–91. Association for Computational Linguistics, 2009.

28.) Ann Taylor, Mitchell Marcus, and Beatrice Santorini. The penn treebank: An overview, 2003.

29.) Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions on, 13(2):260–269, 1967.

30.) Qin Iris Wang and Dale Schuurmans. Improved estimation for unsupervised part-of -speech tagging. In Natural Language Processing and Knowledge Engineering, 2005. IEEE NLP-KE'05. Proceedings of 2005 IEEE International Conference on, pages 219–224. IEEE, 2005.

Appendices

Brown Corpus Tagset

1.		Sentence closer
2	(	Left parenthesis
3	)	Right parenthesis
4	*	*not*, *n't*
5	–	Dash
6	,	Comma
7	:	Colon
8	ABL	Pre-qualifier
9	ABN	Pre-quantifier
10	ABX	Pre-quantifier
11	AP	Post-determiner
12	AT	Article
13	BE	*be*
14	BED	*were*
15	BEDZ	*was*
16	BEG	*being*
17	BEM	*am*
18	BEN	*been*
19	BER	*are*, *art*
20	BEZ	*is*
21	CC	Coordinating conjunction
22	CD	Cardinal numeral
23	CS	Subordinating conjunction
24	DO	*do*
25	DOD	*did*
26	DOZ	*does*
27	DT	Singular determiner
28	DTI	Singular or plural determiner/quantifier
29	DTS	Plural determiner
30	DTX	Determiner/double conjunction
31	EX	Existential
32	FW	Foreign word [4]
33	HL	Word occurring in headline [5]
34	HV	*have*
35	HVD	*had* (past tense)
36	HVG	*having*
37	HVN	*had* (past participle)
38	HVZ	*has*
39	IN	Preposition
40	JJ	Adjective
41	JJR	Comparative adjective
42	JJS	Semantically superlative adjective
43	JJT	Morphologically superlative adjective

44	MD	Modal auxiliary
45	NC	Cited word [5]
46	NN	Singular or mass noun
47	NN$	Possessive singular noun
48	NNS	Plural noun
49	NNS$	Possessive plural noun
50	NP	Proper noun or part of name phrase
51	NP$	Possessive proper noun
52	NPS	Plural proper noun
53	NPS$	Possessive plural proper noun
54	NR	Adverbial noun
55	NRS	Plural adverbial noun
56	OD	Ordinal numeral
57	PN	Nominal pronoun
58	PN$	Possessive nominal pronoun
59	PP$	Possessive personal pronoun
60	PP$$	Second (nominal) possessive pronoun
61	PPL	Singular reflexive/intensive personal pronoun
62	PPLS	Plural reflexive/intensive personal pronoun
63	PPO	Objective personal pronoun
64	PPS	3rd person singular nominative pronoun
65	PPSS	Other nominative personal pronoun
66	QL	Qualifier
67	QLP	Post-qualifier
68	RB	Adverb
69	RBR	Comparative adverb
70	RBT	Superlative adverb
71	RN	Nominal adverb
72	RP	Adverb/particle
73	TL	Word occurring in title [5]
74	TO	Infinitive marker *to*
75	UH	Interjection, exclamation
76	VB	Verb, base form
77	VBD	Verb, past tense
78	VBG	Verb, present participle/gerund
79	VBN	Verb, past participle
80	VBZ	Verb, 3rd person singular present
81	WDT	*wh*-determiner
82	WP$	Possessive *wh*-pronoun
83	WPO	Objective *wh*-pronoun
84	WPS	Nominative *wh*-pronoun
85	WQL	*wh*-qualifier
86	WRB	*wh*-adverb

Penn Treebank Tagset

1	CC	Coordinating conjuction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential *there*
5	FW	Foreign word
6	IN	Preposition/subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol (mathematical or scientific)
25	TO	*to*
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund/present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	*wh*-determiner
34	WP	*wh*-pronoun
35	WP$	Possessive *wh*-pronoun
36	WRB	*wh*-adverb
37	#	Pound sign
38	$	Dollar sign
39	.	Sentence-final punctuation
40	,	Comma
41	:	Colon, semi-colon
42	(	Left bracket character
43	)	Right bracket character
44	"	Straight double quote
45	‘	Left open single quote
46	“	Left open double quote
47	’	Right close single quote
48	”	Right close double quote

HMM Introduction

The goal of this Appendix is to introduce the construction and use of HMMs through a working example. Note that the example presented does not relate to POS tagging.

Consider a manager of a door-to-door sales company. The manager finds that one of the employees has not been doing well in sales. The manager wants to develop a model to determine whether the employee is genuinely struggling to sell products or if the employee is not going door-to-door during the shift. If the employee is not going door-to-door, the employee will be terminated. However, if the sales are down even though the employee is going door-to-door, the employee will not be terminated. To generalize the possibilities, let's say there are only three things the employee could be doing during the shift: going door-to-door and successfully selling products, going door-to-door and not successfully selling products, or not going door-to-door. We will refer to these possibilities as a set of states, $\Sigma = \{s_0, s_1, s_2\}$, respectively. The manager wants to be able to determine if the employee is not going door-to-door. However, the manager cannot simply ask whether or not the employee is going door-to-door, as the employee would probably lie to prevent termination. Therefore, the only thing the manager can do is attempt to make observations about the employee's behavior and appearance to determine if the employee is going door-to-door. Let's assume that the employee either looks happy, irritated, or nervous. We will refer to these observations as, $\Gamma = \{c_0, c_1, c_2\}$, respectively.

The goal is for the manager to use observations in order to determine what the employee did during the shift. However, the manager cannot definitively determine what caused the employee's appearance. Therefore, the manager must make reasonable assumptions as to what caused the appearances. The manager can do so by assigning probabilities to the observations, given the actions of the employee during the work shift. These probabilities are known as *emission* or *observation* probabilities, and will be referred to as the set $B = \{b_{jk}\}$, where $j \in \Gamma,\ k \in \Sigma$, such that $b_{jk} = P(c_k \mid s_j)$. That is, given the employee did not successfully sell products door-to-door (s_1), the probability that the employee looks happy (c_0) is $b_{10} = P(c_0 \mid s_1)$. Let's assume the manager assigns the emission probabilities in Table I.

The manager must also consider the likelihood of the employee's state today, given the state yesterday. For instance, the manager needs the probability the employee will go out and successfully sell products today, given that the employee did not go out yesterday. These variables are known as *transition probabilities* and will be referred to as $A = \{a_{ij}\}$, where $i, j \in \Sigma$, such that $a_{ij} = P(s_j \mid s_i)$. So the probability of the employee going door-to-door and not successfully selling products (s_1) today, given that the employee went door-to-door and successfully sold products (s_0) yesterday is $a_{01} = P(s_1 \mid s_0)$. Let's assume the manager assigns the transition probabilities in Table II.

The final thing that needs to be considered is the initial state. Since this decision does not involve using information from the previous days, the manager assigns the probabilities for each state to be equal. We will refer to these *initial state* probabilities as $\Pi = \{\pi_i\}$, where $i \in \Sigma$, such that $\pi_i = 1/3$ for $i \in \Sigma$. The manager can now use all of this information to define a model, called a *Hidden Markov Model*. See Figure 3 for a diagram of the HMM. Further, the manager's model can be defined as the five-tuple $(\Sigma, \Gamma, \Pi, A, B)$:

1. $\Sigma = \{s_0, s_1, s_2\}$ — Set of States
2. $\Gamma = \{c_0, c_1, c_2\}$ — Set of Observations
3. $\Pi = \{\pi_i\},\ \text{where } i \in \Sigma$ — Initial State Probabilities
4. $A = \{a_{ij}\},\ \text{where } i, j \in \Sigma$ — State transition probabilities
5. $B = \{\beta_{ij}\},\ \text{where } j \in \Sigma,\ k \in \Gamma$ — Emission Probabilities

	Successful Selling	Unsuccessful Selling	Not Going
Happy	0.90	0.05	0.14
Irritated	0.02	0.80	0.08
Nervous	0.08	0.15	0.78

Table 1
Emission Probabilities

	Previous Day Successful Selling	Unsuccessful Selling	Not Going
Successful Selling	0.30	0.45	0.25
Unsuccessful Selling	0.50	0.35	0.50
Not Going	0.20	0.20	0.25

Table 2
Transition Probabilities

The manager can now use this model for two purposes [17]:
1. Find out how well the sequence of observations made by the employer match the current HMM.
2. Find the most probable sequence of states that the employee has been in, based on the observations of the employer.

For purpose 1, the Forward Algorithm is used to calculate the forward probabilities (see Section 3.2). These are used to determine how well a current HMM matches the manager's sequence of observations. For purpose 2, the Viterbi Algorithm is used to find the most probable sequence of states (see Section 3.2). This process is the primary focus of unsupervised POS tagging methods. Finally, it is important to note that the Baum-Welch or Forward-Backward Algorithm can be used to generate a good HMM based on observations made by the employer. For more information on the three tasks and a more in-depth introduction to HMMs refer to [17] and [21].

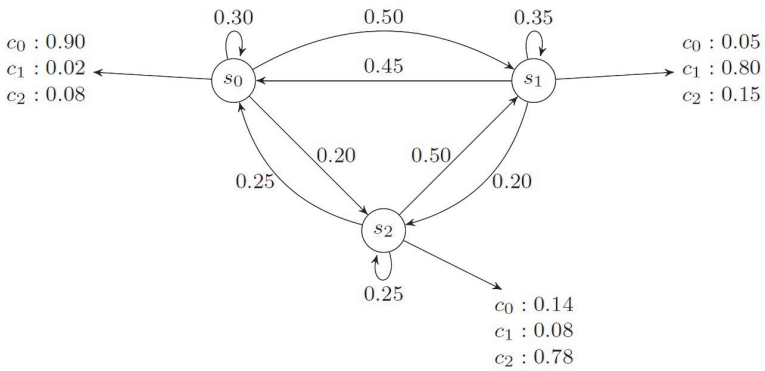

Figure 3
Diagram representing the example HMM with transition and observation probabilities