Building Domain-Specific Taggers without Annotated (Domain) Data

John E. Miller[1] **Manabu Torii**[2] **K. Vijay-Shanker**[1]

[1]Computer & Information Sciences
University of Delaware
Newark, DE 19716
{jmiller,vijay}@cis.udel.edu

[2]Biostatistics, Bioinformatics and Biomathematics
Georgetown University Medical Center
Washington, DC 20057
mt352@georgetown.edu

Abstract

Part of speech tagging is a fundamental component in many NLP systems. When taggers developed in one domain are used in another domain, the performance can degrade considerably. We present a method for developing taggers for new domains without requiring POS annotated text in the new domain. Our method involves using raw domain text and identifying related words to form a domain specific lexicon. This lexicon provides the initial lexical probabilities for EM training of an HMM model. We evaluate the method by applying it in the Biology domain and show that we achieve results that are comparable with some taggers developed for this domain.

1 Introduction

As Natural Language Processing (NLP) technology advances and more text becomes available, it is being applied more and often in specialized domains. Part of Speech (POS) tagging is often a fundamental component to these NLP applications and hence its accuracy can have a significant impact on the application's success. The success that the taggers have attained is often not replicated when the domain is changed. Degradation of accuracy in a new domain can be overcome by developing an annotated corpus for that specific domain, e.g., as in the Biology domain. However, this solution is feasible only if there is sufficient interest in the use of NLP technology in that domain, and there are sufficient funding and resources. In contrast, our approach is to use existing resources, and rapidly develop taggers for new domains without using the time and effort to develop annotated data.

In this work, we use the Wall Street Journal (WSJ) corpus (Marcus et al, 1993) and large amounts of domain-specific *raw* text to develop taggers. We evaluate our methodology in the Biology domain and show the resulting performance is competitive with some taggers built with supervised learning for that domain. Also, we note that the accuracy of taggers trained on the WSJ corpus drops off considerably when applied to this domain. Smith et al. (2005) report that the Brill tagger (1995) has an accuracy of 86.8% on 1000 sentences taken from Medline, and that the Xerox tagger (Cutting et al .1992) has an accuracy of 93.1% on the same sentences. They attribute this drop off to the fact that only 57.8% of the 10,000 most frequent words can be found in WSJ corpus. This observation provides further impetus to developing lexicon for taggers in the new domains.

In the next section, we discuss our general approach. The details of the EM training of the HMM tagger are given in Section 3. Section 4 provides details of how a domain specific lexicon is created. Next, we discuss the evaluation of our models and analysis based on the results. Section 6 discusses related work and those works from which we have taken some ideas. Section 7 has some concluding remarks.

2 Basic Methodology

Inadequate treatment of domain-specific vocabulary is often the primary cause in the degradation of performance when a tagger trained in one genre of text is ported to a new domain. The significance of out-of-vocabulary words has been noted in reduced accuracy of NLP components in the Biology domain (e.g., Lease and Charniak, 2005; Smith et al. 2004). The handling of domain-specific vocabulary is the focus of our approach.

It is quite common to use suffix information in the prediction of POS tags for occurrences of new words. However, its effectiveness may be limited in English, which is not a highly inflected language. However, even for English, we find that not only can suffix information be used online during tagging, but also the presence or absence of morphologically related words can provide considerable information to pre-build a lexicon that associates possible tags with words.

Consider the example of the word "broaden". While the suffix "en" may be utilized to predict the likelihood of verbal tags (VB and VBP) for the word during tagging, if we were to build a lexicon offline, the existence of the words "broadened", "broadening", "broadens" and "broad" give further evidence to treat "broaden" as a verb. This type of information has been used before in (Cucerzan and Yarowsky, 2000).

In the above example, the presence or absence of words with the suffix morphemes suggests POS tag information in two ways: 1) The presence of a suffix morpheme in a word suggests a POS tag or a small set of POS tags for the word. This is the type of information most taggers use to predict tags for unknown words during the tagging process; 2) The presence of the morpheme can also indicate possible tags for the words it attaches to. For example, the derivational morpheme "ment" indicates "government" is likely to be an NN and also that the word it attaches to, "govern" is likely to be a verb. Inflectional and derivational morphemes don't attach to words of just any POS category; they are particular. Thus, we can propose the possibility of JJ (adjective) to "broad" and VB or VBP to "govern" (based on the fact the derivational morphemes "en" and "ment" attach to them) even though by themselves they don't have any suffix information that might be indicative of JJ and VB or VBP.

Additional suffixes (that may or may not be taken from a standard list of English inflectional and derivational morphemes) can also be used. As an example, the suffix "ate" can be associated with a small set of tags: VB or VBP ("educate", "create"), JJ ("adequate", "appropriate"), and NN ("candidate", "climate"). Note the possibility or impossibility of the addition of "tion" and "ly" can help distinguish between the verbal and adjectival situations. In contrast, most taggers that use just suffix information during the tagging process will need strong contextual information (i.e., tags of nearby words) in making their prediction for each occurrence, as such suffixes can be associated with multiple tags.

To utilize such information, we need a dictionary of words in the domain for which we are interested in building a tagger. Such a dictionary will allow us to propose possible tags for a domain word such as "phosphorylate". If we can verify whether words like "phosphorylation", "phosphorylates", and "phosphorylately," are available in the domain then we can obtain considerable information regarding the possible tags that can be associated with "phosphorylate". But we cannot assume the availability of a dictionary of words in the domain. However, it would suffice to have a large text corpus, which we call ***Text-Lex***. We use it as a proxy for a domain dictionary by obtaining a list of words and their relative frequency of appearance in the domain.

Rather than using manually developed rules that assign possible tags for words based on the presence or absence of related words, we wish to apply a more empirical methodology. Since this sort of information is specific to a language rather than a domain, we can use an annotated corpus in another domain to provide exemplars. We use the WSJ (Marcus et al. 1993) corpus, a POS annotated corpus, for this purpose. For example, we can see that "phosphorylate" in the Biology domain and "create" in the WSJ corpus are similar in the sense both take on "tion", "ed", and "ing" suffixes but not "ly" for instance. Since the WSJ corpus would provide POS tag information for "create", we can use it to inform us for "phosphorylate".

The above method forms the basis for our determination of the set of tags that are to be associated with the domain words. However, the actual tag to be assigned for an occurrence in text depends on the context of use. We capture this information by using a first-order HMM tagger model. For the transitional probabilities, we begin by using WSJ-based probabilities as a starting point and then adjust to the new domain by using a domain specific text and using EM training. EM also allows for adjusting lexical probabilities derived using WSJ words as exemplars. We call the domain specific text used for training of our HMM tagger as ***Text-EM***. While this could be the same as ***Text-Lex***, we distinguish the two since ***Text-EM*** could be smaller than ***Text-Lex***. From ***Text-Lex***, we only extract a list of words and their frequency of occurrences. In contrast, we use ***Text-EM*** as a text and hence as a sequence of words.

In this work, the set of suffixes that we use is adapted those found in a GRE preparation web-page (DeForest, 2000). A few additional suffixes were obtained from the online English Dictionary AllWords.com (2005). In the future we expect to consider automatic mining of useful suffixes from a domain. Furthermore, prefixes are also useful for our purposes. However apart from a few prefixes used in hyphenated words, we haven't yet incorporated prefix information in a systematic way into our framework.

In this paper, our evaluation domain is molecular biology. Large amounts of text are easily available in the form of Medline abstracts. We use only about 1% of the Medline text database for ***Text-Lex***. Another reason for selecting this evaluation domain is that we have a considerable amount POS-annotated text in this domain, and the most recent techniques of supervised POS tag learning have been used in developing taggers for this domain. This allows us to evaluate our tagger using the annotated text for evaluation as well as to compare our tagger with others developed for this domain. The POS-annotated text we use is the well-known GENIA (Tateisi et al, 2003) corpus that was developed at University of Tokyo.

3 Expectation Maximization Training

Our tagger is a first-order Hidden Markov Model (HMM) tagger that is trained using Expectation Maximization (EM) since we do not assume existence of annotated data in the new domain.[1] Although we use the GENIA corpus, we take only the raw text and strip off the annotated information for obtaining the ***Text-EM***. Our HMM is based on bigram modeling and hence our transitional probabilities correspond to P(t | t') where t and t' are POS tags. The emissions that label the transition edges will be discussed in the next section and include domain words as well as certain types of "coded words".

[1] We considered a 2nd order model as well, but early work showed negligible advantage predicting to the same training set. Following Wang and Schuurmans (2005) we chose to focus on quality of estimation over model complexity.

The initial transitional probabilities are not randomly chosen but rather taken from the WSJ corpus. If we take the transitional probabilities as a representation of syntactic preferences, then EM learning using ***Text-EM*** may be taken as adjustment of the grammatical preferences in the WSJ corpus to those in the new domain. In order to adjust the grammatical preference to the new domain, we start from smoothed WSJ bigram probabilities. If we started from unsmoothed WSJ bigram probabilities, then EM would not allow us to account for transitions that are not observed in the WSJ corpus. For example, in scientific text, transition from RRB (the right round bracket) to VBG may be possible, while it does not occur in the WSJ corpus. Hence, we smooth the WSJ bigram probabilities with WSJ unigram probabilities.

We compute smoothed initial bigram probabilities as

$$P(t \mid t') = \lambda\, P_{WSJ}(t \mid t') + (1-\lambda)\, P_{WSJ}(t),$$

where λ=0.9. We felt employing techniques suggested in (Brants, 2000) gave too high a preference for unigram probabilities.

The initial emit probability is obtained from the domain text ***Text-Lex***. The process is described in the next section. This information is derived purely from suffix and suffix distribution, or from orthographic information and does not account for the actual context of occurrences in the domain text. We take this suffix-based (and orthographic-based) emit probabilities as reasonable initial lexical probabilities. EM training will adjust them as necessary.

We made one minor modification to the standard forward-backward EM algorithm. We dampen the change in transitional and emit probabilities for each iteration. Significant differences in lexical probabilities between the new domain and WSJ can make undue changes in transitional probabilities and this in turn can further lead the lexical probabilities to head in the wrong direction. By adding a damping factor, we can prevent the unsupervised training to spiral out of control. Hence we let the new transitional probability be given by

$$P(t \mid t') = \lambda\, P_{NEW}(t \mid t') + (1-\lambda)\, P_{OLD}(t \mid t')$$

where P_{OLD} represents the transitional probability in the previous iteration and P_{NEW} represents the probability by standard use of forward-backward algorithm. We use a damping factor of 0.5 for both transitional and emit probabilities. For the emit probabilities, this has the effect of moderating POS preferences derived from the training data and preserving words and POSes from the lexicon for use in the test set.

Even with the damping factor, EM learning followed the pattern of 'Early Maximum' described by Elworthy (1994), where with good initial estimates EM learning only improves accuracy for a few iterations. For our EM training, we fixed iteration 2 as our 'best' EM trained model.

4 Development of the Lexicon and Initial Probabilities

As noted earlier, we use a domain text, ***Text-Lex***, to develop the initial lexical probabilities for the HMM. The essential process is as follows. Let a word w appear a sufficient number of times in ***Text-Lex*** (at least 5 times). We look in ***Text-Lex*** for related words in order to assign a feature vector with this word. Each feature is written as –x+y, where x and y represent suffixes or the empty string (here represented as _).

Features: The feature –x+y represents the word formed by replacing some suffix x in w by some suffix y. Consider the word "creation". "–ion+_" corresponds to the stem word "create" and "–ion+ion" corresponds to the word "creation" itself. The feature "–ion+ed" captures information about the word "created" whereas the feature "-_+s" corresponds to word "creations".

Now consider a word like "history". While this might have non-zero values for "-y+ic" (historic) or "-_+s" (histories), we are likely to set zero value for "–ory+_" (unless "hist" or "histe" is found in ***Text-Lex***). This zero value represents the fact that although "history" has "ory" as a suffix, it has no stem. Such a distinction (whether or not there is a stem) bears much information for suffixes like "ate" and "ory".

We use suffix classes rather than actual suffixes as we believe this provides a more appropriate level of abstraction. Given a word w with a suffix x (for a word with no suffix from our list of suffixes, x is taken to be _. i.e., empty string), we examine whether removal of x from w leads to another word by using a few basic variations that can be found in any rudimentary exposition on English morphology. For example, for the suffix *"ed*", we attempt to replace *"ied"* with *"y"* which relates *"*purified*"* with *"*purify*"* and recognizes the spelling alternation of i/y. Thus for the word *"*purify*"* the feature *"*-+ed" represents the presence of *"*purified*"* since "+ed" represents the suffix class rather than the actual suffix. Similarly, we also consider removal of a suffix and, if necessary, adding an "e" to see if such a word exists. This allows us to relate *"*creation*"* with *"*create*"* or *"*activate*"* with *"*active*"*. Also doubling of a few consonants is attempted to relate *"*occurrence*"* and *"*occur*"*. Finally, when a word could have two suffixes, the word is considered to always have the longer functional suffix. Hence, we consider *"*government*"* to have "ment" suffix rather than "ent" suffix.

Feature Vectors: There are two different types of vectors we use for any word, one called ***Bin*** (for binary count) and other called ***RFreq*** (for relative frequency). In the ***Bin*** vector associated with *"creation",* all these four features will get the value one, assuming that the four corresponding words are found in ***Text-Lex***. On the other hand, assuming *"creatory"* is not found in ***Text-Lex***, the feature "-ion+ory" would get a zero value.

For ***RFreq*** vector, instead of ones and zeros, we first start with the frequency of occurrences of each word and then normalize so that the sum of all feature values is one. Thus, for example, a word with 4 features having non-zero frequencies of 10, 20, 30 and 40 will have the respective values set to 0.1, 0.2, 0.3 and 0.4. A word with four features having non-zero frequency, which are 1, 2, 3 and 4, will also have same 4 relative frequency values.

Our intuition is that the ***Bin*** vector is helpful in determining the set of tags that can be associated with a word and that the ***RFreq*** vector can augment this information regarding the likelihood of these tags. For example, a one for the "-ing+_" feature in a ***Bin*** vector (thus disqualifying a word like "during") may be sufficient to predict VBG, JJ and NN tags. However, this may not suffice to provide the ordering of likelihood among these tags for this word. On the other hand, it seems to be the case that when the "ing" form appears far more often than the "ed" form, then the NN tag is most likely. But if the "ed" form is more frequent, then VBG is most likely. Examples in the WSJ corpus include "smoking", "marketing", "indexing", and "restructuring" for the first kind, and "calling", "counting", "advising", and "noting" for the second kind.

Exemplars in WSJ: Given a word w from ***Text-Lex***, we look for similar words from the WSJ corpus. Even though the set of words used in this corpus may differ substantially from the domain text, our hypothesis is that words with similar suffix distribution will have similar POS tag assignments regardless of the domain. We follow Cucerzan and Yarowsky (2000) in using the kNN method for finding similar words, but we differ in details of the construction of the feature vectors and distance computation. For the word w we create the ***Bin*** and ***RFreq*** vectors based on distribution of words in ***Text-Lex***. Following the same method, we create the ***Bin*** and ***RFreq*** vectors for a word v in the WSJ corpus by using the distributions in the WSJ corpus. Then we compute ***BinDist***(w,v) as the number of features in which the two ***Bin*** vectors differ. A similar ***RFDist*** is defined as a weighted sum of two distances: the first distance is L1-norm distance based on values of features for which both words have non-zero values for and the second distance is based on values of features for which one word has a zero value and other does not. Thus, if the two words' ***RFreq*** vectors are $< w_1,...,w_n >$ and $< v_1,...,v_n >$ respectively then

$$RFsame(w,v) = \sum_{w_i \neq 0 \cap v_i \neq 0} |w_i - v_i|$$

$$RFdiff(w,v) =$$

$$\sum_{w_i = 0 \cap v_i \neq 0} |w_i - v_i| + \sum_{w_i \neq 0 \cap v_i = 0} |w_i - v_i|$$

and,

$$RFDist(w,v) = RFsame(w,v) + \delta RFdiff(w,v)$$

For RFDist(.), we used δ =2. Given a word w, we find the 5 nearest neighbors from the WSJ corpus and use their average lexical probabilities to obtain the lexical probabilities for w. We investigate the use of ***Bin*** vector information and ***RFreq*** vector information for computing the distances (i.e., ***BinDist***(.) and ***RFDist***(.)) as well as a hybrid measure that combines these two distances.

We also considered smoothing the lexical probabilities obtained in the above fashion. Let w be a word for which the above method suggests tags $t_1,\ldots,t_n$ in order of likelihood (t_1 is most probable). Then we consider sqrt-score(t_i)= $\sqrt{n+1-i}$. We then assign probabilities based on this score after normalizing them so that the probabilities for the n tags will sum to 1. Thus, for example, if a word w has three possible tags, no matter what the original lexical probabilities were determined to be, if t_1 is determined to be most probable, then $P(t_1|w)$ will be 0.418 by this method. The second most probable tag will be assigned 0.341.

The intuition behind this square root smoothing method is that this smoothing may be appropriate for low frequency words, where empirical probabilities based purely on a kNN basis may not be entirely appropriate if the new domain is very different. The drawback of course is that if there is sufficient information, we lose useful information by such flattening. And when a tag is significantly more probable for a word then we lose this vital information. For example, the word "high" is mostly annotated as JJ in WSJ corpus but RB and NN are also possible. Square root smoothing will flatten this distribution considerably. Nevertheless, we wish to investigate whether this method of smoothing the distribution is enough in conjunction with EM. EM adjusts the probability from observing the number and context of occurrences in the domain text.[2]

Coded Words: No matter how large ***Text-Lex*** is, there will be words that do not appear a sufficient number of times (we take this number to be 5). We aggregate such words according to their suffixes, if they correspond to one of the predefined suffixes. Then each word with suffix x is considered to be an instance of a "coded" word ***SFX-x***. If a word does not have any of these suffixes then they fall into the coded class ***unknown***. For each such coded word, we assign the tags and probabilities based on similarly aggregated words in the WSJ corpus.

We have two other broad classes of words that we treat differently. Coded words are formed based on orthographic characteristics, which include but are not limited to Greek letters, Roman numerals, digits, upper or lower case single letters, upper case letter sequences, cardinals, certain prefix words, and their combinations. Since they are relatively easy to tag, we do not use the WSJ corpus for them but handle it programmatically. Finally, if a word occurs often in WSJ or is assigned tags such as CD, FW, MD, PRP, DT, WDT, etc. (tags which can't be predicted by means of suffix or suffix-related words), we add this word together with the tags and probability into the domain lexicon that we are building.

[2] We also considered linear and square functions for smoothing while reporting only the sqrt results in section 5.

5 Evaluation and Analysis

As noted earlier, our evaluation is on molecular biology text. For ***Text-Lex***, we used 133,666 titles/abstracts of research papers, a small fraction of the Medline database available from the National Library of Medicine. These abstracts were contained in just five of the 500 compressed data files in the 2006 version of the Medline database. These abstracts cover topics more broadly in Biomedicine and not just molecular biology. On the other hand, we use for ***Text-EM***, text which can be regarded to be in a subfield of molecular biology.

Text-EM is the text from the GENIA corpus (version 3.02) described in (Tateisi et al. 2003). This corresponds to about 2000 abstracts, which are annotated with POS tag information (using the same tags used in the WSJ corpus). We use a 5-fold cross-validation, i.e., 5 partitions are formed and experiments conducted 5 times and results averaged. For each test partition, the remainder partitions are used for "training". In our case, this is unsupervised since we use EM and hence we totally disregard the POS tag information that is associated with the words. We note that both the text for EM training as well as for testing come from the same domain.

We first evaluate the process of building the lexicon. This time we consider the entire GENIA corpus and not any partition. We first considered all words in the GENIA corpus for which we can expect our kNN method to assign a tag. Hence all words that would be treated as coded words are ignored. For each such word, we consider the tags assigned to them in the GENIA corpus and form pairs <w,t>. We are interested in the word type and not token and hence we will not have any multiple occurrences of a pair <w,t>. Our kNN method identifies 96.3% of these pairs; we can think of this as recall. This makes our approach effective, especially given the fact that the kNN method only assigns 1.92 tags on an average to these words in the GENIA corpus. Next considering all words appearing in the GENIA corpus, our lexicon includes a correct tag in 99.0% of the cases on a word-token basis. These results are summarized below.

Characteristic	**Statistic**
kNN Recall (word-type)	96.3%
Average Number Tags/Word	1.92 tags
Lexicon Recall (word-token)	99.0%

We now turn to the evaluation of the accuracy of our HMM. As mentioned earlier, these results are based on 5-fold cross-validation experiments. The best results (95.77%) were obtained for the case where we took the lexical probabilities directly from kNN using only ***RFDist*** and by discarding all tags assigned with probability less than 0.02.[3]

These results compare favorably to other taggers developed for the Biology domain. The MedPost tagger (see Section 6) achieved an accuracy of 94.1% when we applied it to the GENIA abstracts. The PennBioIE tagger (see Section 6) achieved an accuracy of 95.1%. Note that output from the PennBioIE tagger is not fully compatible with GENIA annotation due to some differences in its tokenization. Even if the differences in accuracies can be discounted due to tokenization or even systematic differences in annotation between the training and test corpora, our main point is that our results compare favorably (our tagger competitive) with taggers that were developed for the Biomedicine domain using supervised training.

These results are summarized in the table below.

POS Tagger	**%Accuracy**
Our HMM (5-fold)	95.77%
MedPost	94.1%
PennBioIE	95.1%
GENIA supervised	98.26%

MedPost seems intended to cover all of Biomedicine, since its lexicon is based on the 10,000 most frequently occurring words from Medline and for which the set of possible tags were manually specified. The PennBioIE tagger was developed using 315 Medline abstracts using another subfield of molecular biology.

None of these accuracies however are as high as those of the GENIA tagger (Tsuruoka et al. 2005) which was trained (supervised) using GENIA corpus and uses a machine learning model more sophisticated than the simple first-order HMM tagger we use. This model considers more features including words to the right. The best results (98.26%) were obtained when lexicon from three different sources were aggregated.

[3] Banko and Moore (2004) showed only slight improvement in tag accuracy between .01 and .1 cutoffs with a lexicon built from annotated data. We opted for the .02 cutoff because of our 'noisier' lexicon.

Returning to the results for our taggers, we also tried ***BinDist*** in the kNN method, with and without square root smoothing. These results were typically less than the above-mentioned result. We also compared using a square root smooth on ***RFDist*** obtaining results approximately 1% lower than without the square root smooth.

We next present some examples that illustrate strengths and weaknesses of the current model. An example that shows that EM training makes good adjustment to the domain is the improvement in tagging of verbal categories. We conducted a detailed error analysis on one of the cross-validation partitions and noted that the accuracy on all verbal POS tags improved after EM training. A noteworthy case is the improvement in tagging of VBP originally misclassified as VB. Since most English words that are VB can also be VBP, and since they are annotated more frequently in WSJ as VB, the initial lexicon usually has a higher probability assigned to VB for most words. As EM training progresses, we noted that the frequency of VBP mistagged as VB decreases. Similarly, misclassifications of VBG as NN also drops in the final model (by 40.3% on ***Text-EM***) as compared to the initial model based on WSJ transitional probabilities and initial lexicon derived using WSJ words as exemplars.

Previously, in the context of parsing Biomedical text, Lease and Charniak (2004) mention the occurrences of sequences of multiple NN is more frequent in the GENIA corpus than in the WSJ corpus and that it could lead to parsing errors. We didn't observe this problem here, but rather the contrary situation where many JJs were initially mistagged as NN. About 22% of these misclassifications are corrected after EM training.

While our model adjusts well in these cases to the new domain, sometimes the drift leads to worse performance. An example is in the misclassification of VBN as JJ. The most frequent word for which this misclassification occurs in the word "activated". These misclassifications occur in the context such as "the activated cells". The use of VBN rather than JJ is hard to determine on basis of just surface features and perhaps has to do more with the meaning of the word. In supervised setting, if sufficient such cases were annotated then this would be learned. But in an unsupervised setting this turns out to be a problem case. Despite the fact that ***RFDist*** predicted VBN as most probable tag for "activated", EM training makes this situation worse.

Analysis of words with most frequent errors revealed many cases from orthographic coded words. Many occurrences of single lower case letters (which could have LS, SYM or NN tags) were labeled as LS whereas the GENIA tagging used NN. Our model tagged "+/-" always as SYM whereas because of the context of use, GENIA annotations were CC. (In fact, GENIA does not appear to use the SYM tag.) Similarly, "<" and ">" were often mistagged as SYM by our model whereas based on context they are annotated as JJR.

6 Related Work

The impact of out-of-vocabulary words on NLP applications has been noted before. The degradation in performance of components, which were trained on the WSJ corpus, but used on biomedical text has been noted (Lease and Charniak, 2004, Smith et al, 2005). Smith et al. (2005) use this observation in the design of their POS tagger, MedPost, by building a Markov model with a lexicon containing the 10,000 most frequent words from Medline, and using annotated text from the Biomedical text for supervised training.

There are many unsupervised approaches to POS tagging. We focus now on those that are most closely related to our work and contain ideas that have influenced this work. There have been many uses of EM training to build HMM taggers (Kupiec, 1992; Elworthy, 1994; Banko and Moore, 2004; Wang and Schuurmans, 2005). Banko and Moore (2004) achieved better accuracy by restricting the set of possible tags that are associated with words. By eliminating possibilities that may appear rarely with a word, they reduce the chances of unsupervised training spiraling along an unlikely path. We believe by using our approach we considerably reduce the set of tags to what is appropriate for each word. Further, we too remove any tag associated with low probability by kNN method. Usually these tags are noise introduced by some inappropriate exemplar.

Wang and Schuurman (2005) suggest that EM algorithm be modified such that at any iteration the unigram tag probability be held constant to the true probability for each tag. Again, this might serve to stop a drift in unsupervised methods towards making a tag's probability become larger than it should be. However, the true probability cannot be known ahead of time and certainly not in a new domain. While a WSJ bigram probability need not reflect the corresponding preferences in the new domain, our use of starting from WSJ probabilities and then damping changes to transition probabilities was motivated by a similar concern of not letting a drift towards making some (bigram) tags too frequent during EM iterations.

Using suffixation patterns for purposes of predicting POS tags has been considered before. Although as far as we know, we are the first to apply it for domain adaptation purposes. Schone and Jurafsky (2001) consider clusters of words (obtained by some "perfect" clustering algorithm) and then compute a measure of how "affixy" a cluster is. For example, a cluster containing words "climb" and "jump" may be related by suffixing operation +s to a cluster that contains words "climbs" and "jumps". The percentage of words in a cluster that are so related provides a measure of how "affixy" a cluster. This together with five other attributes of clusters (such as whether words in a cluster precede those of another cluster, optionality) and language universals induce POS tags for these clusters from corpora. This method does not use POS tagged corpora (although in the reported experiment the initial "perfect" clusters were obtained from the Brown corpus using the POS tag information). In contrast, we use the POS tagged WSJ corpus to assist in the induction of tag information for our lexicon. In this respect, our method is closer to the approach of Cucerzan and Yarowsky (2000). Our use of the kNN method to identify tags and their probabilities for words was inspired by this work. However, their use of kNN method was in the context of supervised learning. The method was applied for handling words unseen in the training data. The estimated probabilities were used during the tagging process. Instead of just applying the method for unknown words, i.e., words not present in the training data, our approach is to create the entire lexicon in the new domain. As Lease and Charniak (2004), among others, have noted, the distribution of NN tag sequences as well as tag distributions in the Biomedical domain could differ from WSJ text. Since our aim is to adjust to the new domain, we employed unsupervised learning in the form of EM training, unlike the supervised tagging model development approach of Cucerzan and Yarowsky. Another significant difference is that their method determines nearest neighbors not only on the basis of suffix-related words but also on the basis of nearby words context. Since our motivation, on the other hand, is to move to a new domain, we didn't consider detection of similarity on the basis of word contexts. In contrast, we have shown that the approach of identifying words on the basis of suffixation patterns and using them as exemplars can be applied effectively even when the domain of application is substantially different from the text (the WSJ corpus) providing the exemplars.

7 Conclusions

As NLP technology continues to be applied in new domains, it becomes more important to consider the issue of portability to new domains. To cope with domain-specific vocabulary and also different use of vocabulary in a new domain, we exploited suffix information of words. While use of suffix information per se has been employed in many existing POS taggers, its use is often limited to an online manner, where each word is examined independently from the existence of its morphologically related words. As shown in (Cucerzan and Yarowsky, 2000), such information can provide considerable information to build a lexicon that associates possible tags with words. However, we use this information only to provide the initial values. We apply EM algorithm to adjust these initial probabilities to the new domain.

The results in Section 5 show that we achieve good performance in the evaluation domain, which is comparable with two recently developed taggers for this domain. We also show in section 5 examples of how EM unlearns some WSJ bias and adjusts to the new domain. While we introduce a damping factor to slow down changes in iterations of EM training, we believe there is scope for further improvement to minimize drift. Furthermore, there is scope to improve our kNN method as discussed at the end of Section 5. In the future, we also expect to consider methods that may automatically mine suffixes in a new domain and use these domain-specific suffixes. We used the kNN method to associate words in the new domain with possible POS tags.

Despite the often-stated notion that English is not morphologically rich, we find that suffix-based methods can still help make significant inroads.

Our method offers the chance to develop good taggers for specialized domains. For example, the GENIA corpus and PennBioIE corpus are specializations within molecular biology, but taggers developed on one corpus degrades in performance on the other. Using our method, we could use different ***Text-EM*** for these specializations even if we retain Medline as ***Text-Lex***. In the same way, we could develop a tagger for the medical domain, which has a distinct vocabulary from biology.

References

Banko, M. and Moore, R.C. 2004. Part of Speech Tagging in Context. In Proceedings, 20th International Conference on Computational Linguistics (Coling 2004), Geneva, Switzerland, pp.556-561.

Baum, L. 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. Inequalities 3:1-8.

Brants, T. 2000. TNT - a statistical part-of-speech tagger. In Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000.

Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. Computational Linguistics, 21(4):543-565.

Cucerzan, S. and Yarowsky, D. 2000. Language independent minimally supervised induction of lexical probabilities. Proceedings of ACL-2000, Hong Kong, pages 270-277.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. 1992. A practical part of sppech tagger. Proceedings of 3rd Conference on Applied Natural Language Processing, 53-58.

DeForest, J. 2000. Graduate Record Exam Suffixed web page. Michigan State University. http://www.msu.edu/~defores1/gre/sufx/gre_suffx.htm

Elworthy, D. 1994. Does Baum-Welch re-estimation help taggers. In Proceedings of the Fourth Conference on Applied Natural Language Processing, ACL.

Kulick, S., Bies, A., Liberman, M., Mandel, M., McDonald, R., Palmer, M., Schein, A. and Ungar, L. *Integrated Annotation for Biomedical Information Extraction.* HLT/NAACL, 2004.

Kupiec, J. 1992. Robust Part-of-speech Tagging Using a Hidden Markov Model. Computer Speech and Language, 6.

Lease, M. and Charniak, E. 2005. Parsing Biomedical Literature. IJCNLP-2005: 58-69.

Marcus, M., Santorini, B., Marcinkiewicz, M.A. 1993. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 19:313-330.

PennBioIE. 2005. Mining The Bibliome Project. http://bioie.ldc.upenn.edu/.

Schone, P. and Jurafsky, D. 2001, Language Independent Induction of Part of Speech Class Labels Using Language Universals. IJCAI Workshop on Text Learning: Beyond Supervision.

Smith, L., Rindflesch, T., Wilbur, W.J. 2004. MedPost: a part-of-speech tagger for biological text. Bioinformatics 20 (14):2320-2321.

Smith, L., Rindflesch, T., Wilbur, W.J. 2005. The importance of the lexicon in tagging biomedical text. Natural Language Engineering 12(2) 1-17.

Tateisi, Y., Ohta, T., dong Kim, J., Hong, H., Jian, S., Tsujii, J. 2003. The GENIA corpus: Medline abstracts annotated with linguistic information. In: Third meeting of SIG on Text Mining, Intelligent Systems for Molecular Biology (ISMB).

Tsuruoka, Y., Tateishi, Y., Kim, J. D., Ohta, T., McNaught, J., Ananiadou, S., and Tsujii, J.. 2005. Developing a Robust Part-of-Speech Tagger for Biomedical Text, Advances in Informatics - 10th Panhellenic Conference on Informatics, LNCS 3746: 382-392.

Wang, Q. and Schuurmans, D. 2005. Improved estimation for unsupervised part-of-speech tagging. In IEEE NLP-KE

www.AllWords.com. 2005. English Dictionary and Language Guide. AllSites LLC. www.AllSitesllc.com