Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging

Eric Brill[1]
Department of Computer Science
Johns Hopkins University
brill@cs.jhu.edu

Abstract

In this paper we describe an unsupervised learning algorithm for automatically training a rule-based part of speech tagger without using a manually tagged corpus. We compare this algorithm to the Baum-Welch algorithm, used for unsupervised training of stochastic taggers. Next, we show a method for combining unsupervised and supervised rule-based training algorithms to create a highly accurate tagger using only a small amount of manually tagged text.

Introduction

There has recently been a great deal of work exploring methods for automatically training part of speech taggers, as an alternative to laboriously hand-crafting rules for tagging, as was done in the past [Klein and Simmons, 1963; Harris, 1962]. Almost all of the work in the area of automatically trained taggers has explored Markov-model based part of speech tagging [Jelinek, 1985; Church, 1988; Derose, 1988; DeMarcken, 1990; Cutting *et al.*, 1992; Kupiec, 1992; Charniak *et al.*, 1993; Weischedel *et al.*, 1993; Schutze and Singer, 1994; Lin *et al.*, 1994; Elworthy, 1994; Merialdo, 1995].[2] For a Markov-model based tagger, training consists of learning both lexical probabilities ($P(word|tag)$) and contextual probabilities ($P(tag_i|tag_{i-1} \ldots tag_{i-n})$). Once trained, a sentence can be tagged by searching for the tag sequence that maximizes the product of lexical and contextual probabilities.

The most accurate stochastic taggers use estimates of lexical and contextual probabilities extracted from large manually annotated corpora (eg. [Weischedel *et al.*, 1993; Charniak *et al.*, 1993]). It is possible to use unsupervised learning to train stochastic taggers without the need for a manually annotated corpus by using the Baum-Welch algorithm [Baum, 1972; Jelinek, 1985; Cutting *et al.*, 1992; Kupiec, 1992; Elworthy, 1994; Merialdo, 1995]. This algorithm works by iteratively adjusting the lexical and contextual probabilities to increase the overall probability of the training corpus. If no prior knowledge is available, probabilities are initially either assigned randomly or evenly distributed. Although less accurate than the taggers built using manually annotated corpora, the fact that they can be trained using only a dictionary listing the allowable parts of speech for each word and not needing a manually tagged corpus is a huge advantage in many situations. Although a number of manually tagged corpora are available (eg. [Francis and Kucera, 1982; Marcus *et al.*, 1993]), training on a corpus of one type and then applying the tagger to a corpus of a different type usually results in a tagger with low accuracy [Weischedel *et al.*, 1993]. Therefore, if tagged text is needed in training, this would require manually tagging

[1] This work was funded in part by NSF grant IRI-9502312.

[2] Some other approaches to tagging are described in [Hindle, 1989; Black *et al.*, 1992].

text each time the tagger is to be applied to a new language, and even when being applied to a new type of text.

In [Brill, 1992; Brill, 1994], a rule-based part of speech tagger is described which achieves highly competitive performance compared to stochastic taggers, and captures the learned knowledge in a set of simple deterministic rules instead of a large table of statistics. In addition, the learned rules can be converted into a deterministic finite state transducer. Tagging with this finite state transducer requires n steps to tag a sequence of length n, independent of the number of rules, and results in a part of speech tagger ten times faster than the fastest stochastic tagger [Roche and Schabes, 1995]. One weakness of this rule-based tagger is that no unsupervised training algorithm has been presented for learning rules automatically without a manually annotated corpus. In this paper we present such an algorithm. We describe an algorithm for both unsupervised and weakly supervised training of a rule-based part of speech tagger, and compare the performance of this algorithm to that of the Baum-Welch algorithm.

Transformation-Based Error-Driven Learning

The rule-based tagger is based on a learning algorithm called transformation-based error-driven learning. Transformation-based error-driven learning has been applied to a number of natural language problems, including part of speech tagging, prepositional phrase attachment disambiguation, speech generation and syntactic parsing [Brill, 1992; Brill, 1994; Ramshaw and Marcus, 1994; Roche and Schabes, 1995; Brill and Resnik, 1994; Huang *et al.*, 1994; Brill, 1993a; Brill, 1993b]. Figure 1 illustrates the learning process. First, unannotated text is passed through an initial-state annotator. The initial-state annotator can range in complexity from assigning random structure to assigning the output of a sophisticated manually created annotator. Once text has been passed through the initial-state annotator, it is then compared to the *truth* as specified in a manually annotated corpus, and transformations are learned that can be applied to the output of the initial state annotator to make it better resemble the *truth*.

In all of the applications explored to date, the following greedy search is applied: at each iteration of learning, the transformation is found whose application results in the *highest score*; that transformation is then added to the ordered transformation list and the training corpus is updated by applying the learned transformation. To define a specific application of transformation-based learning, one must specify the following:

1. The initial state annotator.

2. The space of transformations the learner is allowed to examine.

3. The scoring function for comparing the corpus to the *truth* and choosing a transformation.

Once an ordered list of transformations is learned, new text can be annotated by first applying the initial state annotator to it and then applying each of the learned transformations, in order.

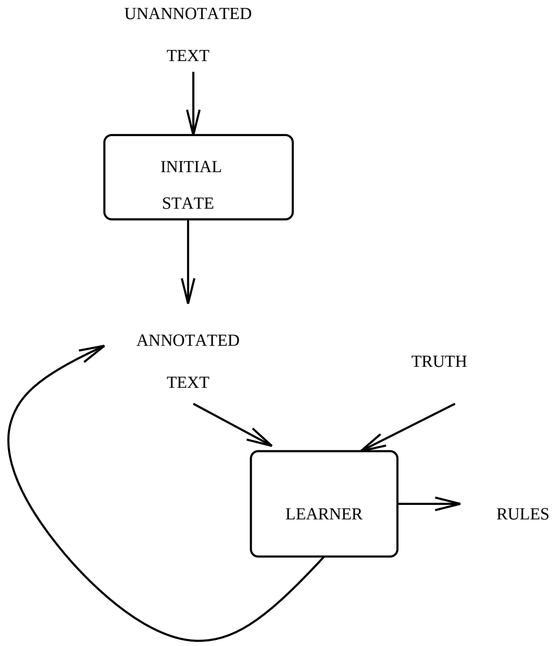

Figure 1: Transformation-Based Error-Driven Learning.

Transformation-Based Part of Speech Tagging

In transformation-based part of speech tagging,[3] all words are initially tagged with their most likely tag, as indicated in the training corpus. Below are some of the transformation templates used by the learner.[4]

Change tag **a** to tag **b** when:

1. The preceding (following) word is tagged z.
2. The preceding (following) word is w.
3. The word two before (after) is w.
4. One of the two preceding (following) words is tagged z.
5. The current word is w and the preceding (following) word is x.
6. The current word is w and the preceding (following) word is tagged z.

The evaluation measure is simply tagging accuracy. In each learning iteration, the system learns that transformation whose application results in the greatest reduction of error.[5] Because the learning algorithm is data-driven, it only needs to consider a small

[3] For a more detailed description of supervised transformation-based part of speech tagging, see [Brill, 1994].

[4] In [Brill, 1994], a total of 21 templates are used.

[5] Note an important difference between Markov-model based taggers and the transformation-based tagger: the former attempts to maximize the probability of a string, whereas the latter attempts to minimize the number of errors.

percentage of all possible transformations when searching for the best one. An example of a learned transformation is:

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

If the word *race* occurs more frequently as a verb than as a noun in the training corpus, the initial state annotator will mistag this word as a verb in the sentence: *The race was very exciting.* The above transformation will correct this tagging error.

It was shown in [Brill, 1994] that the transformation-based tagger achieves a high rate of tagging accuracy. The transformation-based tagger captures its learned information in a set of simple rules, compared to the many thousands of opaque probabilities learned by Markov-model based taggers.[6] Supervised training is feasible when one has access to a large manually tagged training corpus from the same domain as that to which the trained tagger will be applied. We next explore unsupervised and weakly supervised training as a practical alternative when the necessary resources are not available for supervised training.

Unsupervised Learning of Transformations

In supervised training, the corpus is used for scoring the outcome of applying transformations, in order to find the best transformation in each iteration of learning. In order to derive an unsupervised version of the learner, an objective function must be found for training that does not need a manually tagged corpus.

We begin our exploration providing the training algorithm with a minimal amount of initial knowledge, namely knowing the allowable tags for each word, and nothing else.[7] The relative likelihoods of tags for words is not known, nor is any information about which tags are likely to appear in which contexts. This would correspond to the knowledge that could be extracted from an on-line dictionary or through morphological and distributional analysis.

The unsupervised rule learning algorithm is based on the following simple idea. Given the sentence:

The **can** will be crushed.

with no information beyond the dictionary entry for the word **can**, the best we can do is randomly guess between the possible tags for **can** in this context. However, using an unannotated corpus and a dictionary, it could be discovered that of the words that appear after **The** in the corpus that have only one possible tag listed in the dictionary, nouns are much more common than verbs or modals. From this the following rule could be learned:

Change the tag of a word from **(modal OR noun OR verb)** to **noun** if the previous word is **The**.

[6] The transformation-based tagger is available through anonymous ftp to ftp.cs.jhu.edu in /pub/brill/Programs.

[7] In this paper we ignore the problem of unknown words: words appearing in the test set which did not appear in the training set. We plan to explore ways of processing unknown words in future work, either by initially assigning them all open-class tags, or devising an unsupervised version of the rule-based unknown word tagger described in [Brill, 1994].

To fully define the learner, we must specify the three components of the learner: the initial state annotator, the set of transformation templates, and the scoring criterion.

Initial State Annotator The unsupervised learner begins with an unannotated text corpus, and a dictionary listing words and the allowable part of speech tags for each word. The tags are not listed in any particular order. The initial state annotator tags each word in the corpus with a list of all allowable tags. Below is an example of the initial-state tagging of a sentence from the Penn Treebank [Marcus *et al.*, 1993], where an underscore is to be read as **or**.[8]

Rival/JJ_NNP gangs/NNS have/VB_VBP turned/VBD_VBN cities/NNS into/IN combat/NN_VB zones/NNS ./.

Transformation Templates The learner currently has four transformation templates. They are:

Change the tag of a word from χ to Y if:

1. The previous tag is T.
2. The previous word is W.
3. The next tag is T.
4. The next word is W.

Transformations are used differently in the unsupervised learner than in the supervised learner. Here, a transformation will reduce the uncertainty as to the correct tag of a word in a particular context, instead of changing one tag to another. So all learned transformations will have the form:

Change the tag of a word from χ to Y in context C

where χ is a set of two or more part of speech tags, and Y is a single part of speech tag, such that $Y \in \chi$. Below we list some transformations that were actually learned by the system.

Change the tag:

From NN_VB_VBP to VBP if the previous tag is NNS
From NN_VB to VB if the previous tag is MD
From JJ_NNP to JJ if the following tag is NNS

[8] JJ= Adjective, MD = Modal, NNP = Singular Proper Noun, NN = Singular or Mass Noun, POS = Possessive, VB = Verb, Base Form, VBD = Verb, Past Tense, VBN = Verb, Past Part., VBP = Verb, Non-3rd Person Sing. Present.

Scoring Criterion When using supervised transformation-based learning to train a part of speech tagger, the scoring function is just the tagging accuracy that results from applying a transformation. With unsupervised learning, the learner does not have a gold standard training corpus with which accuracy can be measured. Instead, we can try to use information from the distribution of unambiguous words to find reliable disambiguating contexts.

In each learning iteration, the score of a transformation is computed based on the current tagging of the training set. Recall that this is completely unsupervised. Initially, each word in the training set is tagged with all tags allowed for that word, as indicated in the dictionary. In later learning iterations, the training set is transformed as a result of applying previously learned transformations. To score the transformation: *Change the tag of a word from χ to Y in context C*, where $Y \in \chi$, we do the following. For each tag $Z \in \chi$, $Z \neq Y$, compute

$$freq(Y)/freq(Z) * incontext(Z, C)$$

where freq(Y) is the number of occurrences of words unambiguously tagged with tag Y in the corpus, freq(Z) is the number of occurrences of words unambiguously tagged with tag Z in the corpus, and incontext(Z,C) is the number of times a word unambiguously tagged with tag Z occurs in context C in the training corpus.[9]

Let

$$R = argmax_Z \; freq(Y)/freq(Z) * incontext(Z, C)$$

Then the score for the transformation *Change the tag of a word from χ to Y in context C* is:

$$incontext(Y, C) - freq(Y)/freq(R) * incontext(R, C)$$

A good transformation for removing the part of speech ambiguity of a word is one for which one of the possible tags appears much more frequently as measured by unambiguously tagged words than all others in the context, after adjusting for the differences in relative frequency between the different tags. The objective function for this transformation measures this by computing the difference between the number of unambiguous instances of tag Y in context C and the number of unambiguous instances of the most likely tag R in context C, where $R \in \chi$, $R \neq Y$, adjusting for relative frequency. In each learning iteration, the learner searches for the transformation which maximizes this function. Learning stops when no positive scoring transformations can be found.

Unsupervised Learning: Results

To test the effectiveness of the above unsupervised learning algorithm, we ran a number of experiments using two different corpora and part of speech tag sets: the Penn Treebank Wall Street Journal Corpus [Marcus *et al.*, 1993] and the original Brown Corpus [Francis and Kucera, 1982]. First, a dictionary was created listing all possible tags for each word in the corpus. This means that the test set contains no unknown words. We have set up the experiments in this way to facilitate comparisons with results given in other papers, where the same was done.

[9] An example of a context is: the previous tag is a determiner.

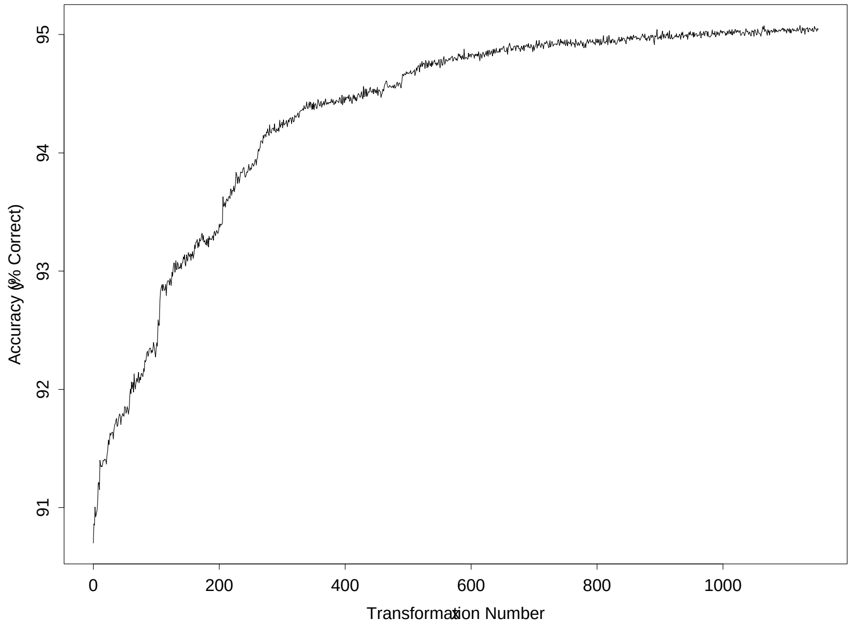

Figure 2: Test Set Accuracy vs Transformation Number for the Penn Treebank Wall Street Journal Corpus

Penn Treebank Results

In this experiment, a training set of 120,000 words and a separate test set of 200,000 words were used. We measure the accuracy of the tagger by comparing text tagged by the trained tagger to the gold standard manually annotated corpus. In the case where the tag of a word is not fully disambiguated by the tagger, a single tag is randomly chosen from the possible tags, and this tag is then compared to the gold standard. Initial state tagging accuracy on the training set is 90.7%. After learning 1,151 transformations, training set accuracy increases to 95.0%. Initial state tagging accuracy on the test set is also 90.7%. Accuracy increases to 95.1% after applying the learned transformations.

Figure 2 shows test set tagging accuracy as a function of transformation number. In figure 3, we plot the difference between training and test set accuracies after the application of each transformation, including a smoothed curve.[10] Notice that there is no overtraining: the difference in accuracies on training and test set remain within a very narrow range throughout, with test set accuracy exceeding training set accuracy by a small margin. Overtraining did not occur when using the original Brown Corpus either. When training a stochastic tagger using the Baum-Welch algorithm, overtraining often does occur [Merialdo, 1995; Elworthy, 1994], requiring an additional held-out training corpus for determining an appropriate number of training iterations.

[10] The graphs are choppy because after each transformation is applied, correctness for words not yet fully disambiguated is judged after randomly selecting from the possible tags for that word.

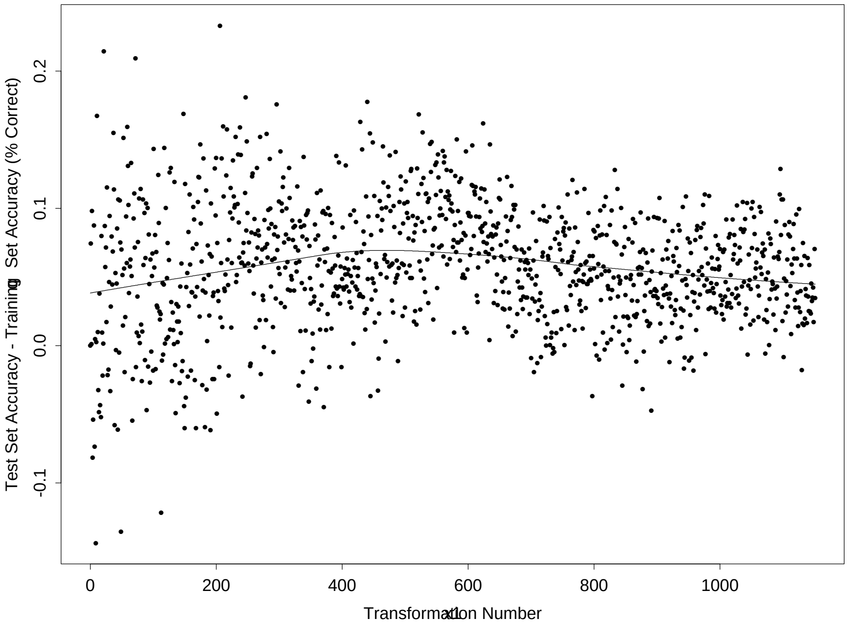

Figure 3: Difference Between Training and Test Set Accuracies.

Corpus	Training Corpus Size (Words)	% Correct
Penn Treebank	120K	95.1
Brown Corpus	120K	95.6
Brown Corpus	350K	96.0

Table 1: Unsupervised Training: Test Set Accuracy

Brown Corpus Results

In this experiment, we also used a training set of 120,000 words and a separate test set of 200,000 words. Initial state tagging accuracy on the training set is 89.8%. After learning 1,729 transformations and applying them to the training set, accuracy increases to 95.6%. Initial state tagging accuracy on the test set is 89.9%, with accuracy increasing to 95.6% after applying the learned transformations. Expanding the training set to 350,000 words and testing on the same test set, accuracy increases to 96.0%. All unsupervised learning results are summarized in table 1.

Comparison With Other Results

In [Merialdo, 1995], tagging experiments are described training a tagger using the Baum-Welch algorithm with a dictionary constructed as described above and an untagged corpus. Experiments were run on Associated Press articles which were manually tagged at the University of Lancaster. When training on one million words of text, test set accuracy peaks at 86.6%. In [Elworthy, 1994], similar experiments were run. There, a peak accuracy of 92.0% was attained using the LOB corpus.[11] Using the Penn Treebank corpus, a peak accuracy of 83.6% resulted. These results are significantly lower than the results achieved using unsupervised transformation-based learning.

In [Kupiec, 1992] a novel twist to the Baum-Welch algorithm is presented, where instead of having contextual probabilities for a tag following one or more previous tags, words are pooled into equivalence classes, where all words in an equivalence class have the same set of allowable part of speech assignments. Using these equivalence classes greatly reduces the number of parameters that need to be estimated. Kupiec ran experiments using the original Brown Corpus. When training on 440,000 words, test set accuracy was 95.7%, excluding punctuation. As shown above, test set accuracy using the transformation-based algorithm described in this paper gives an accuracy of 96.0% when trained on 350,000 words. Excluding punctuation, this accuracy is 95.6%. Note that since the Baum-Welch algorithm frequently overtrains, a tagged text would be necessary to figure out what training iteration gives peak performance.

Weakly Supervised Rule Learning

We have explored a method of training a transformation-based tagger when no information is known other than a list of possible tags for each word. Next we explore weakly supervised learning, where a small amount of human intervention is permitted. With Markov-model based taggers, there have been two different methods proposed for adding knowledge to a tagger trained using the Baum-Welch algorithm. One method is to manually alter the tagging model, based on human error analysis. This method is employed in [Kupiec, 1992; Cutting *et al.*, 1992]. Another approach is to obtain the initial probabilities for the model directly from a manually tagged corpus instead of using random or evenly distributed initial probabilities, and then adjust these probabilities using the Baum-Welch algorithm and an untagged corpus. This approach is described in [Merialdo, 1995; Elworthy, 1994].

A tagged corpus can also be used to improve the accuracy of unsupervised transformation-based learning. A transformation-based system is a processor and not a classifier. Being a processor, it can be applied to the output of any initial state annotator. As mentioned above, in the supervised transformation-based tagger described in [Brill, 1994], each word is initially tagged with its most likely tag. Here, we use the trained unsupervised part of speech tagger as the initial state annotator for a supervised learner. Transformations will then be learned to fix errors made by the unsupervised learner. As shown in figure 4, unannotated text is first passed through the unsupervised initial-state annotator, where each word is assigned a list of all allowable tags. The output of this tagger is then passed to the unsupervised learner, which learns an ordered list of transformations. The initial-state annotator and learned unsupervised transformations are then applied to unannotated text, which is then input to the supervised learner, along with the corresponding manually tagged corpus. The supervised learner learns a second ordered list of transformations.

Once the system is trained, fresh text is tagged by first passing it through the unsupervised initial state annotator, then applying each of the unsupervised transformations, in order, and then applying each of the supervised transformations, in order.

The advantage of combining unsupervised and supervised learning over using supervised

[11] [Elworthy, 1994] quotes accuracy on ambiguous words, which we have converted to overall accuracy.

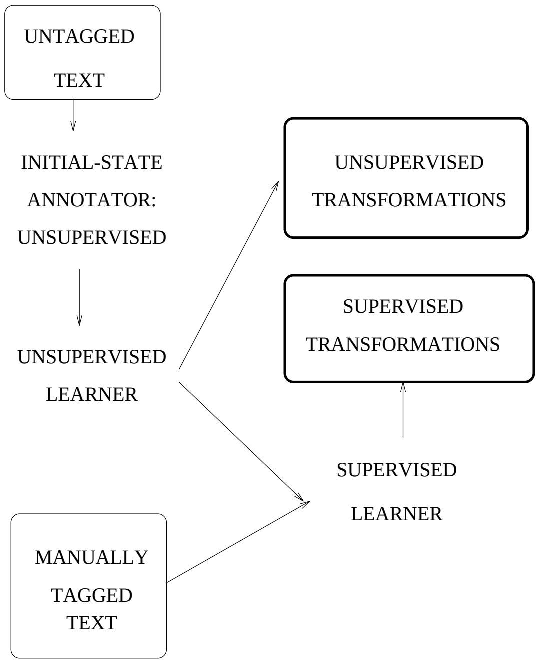
UNTAGGED TEXT
INITIAL-STATE ANNOTATOR: UNSUPERVISED
UNSUPERVISED LEARNER
UNSUPERVISED TRANSFORMATIONS
SUPERVISED TRANSFORMATIONS
SUPERVISED LEARNER
MANUALLY TAGGED TEXT

Figure 4: Combining Unsupervised and Supervised Learning

learning alone is that the combined approach allows us to utilize both tagged and untagged text in training. Since manually tagged text is costly and time-consuming to generate, it is often the case that when there is a corpus of manually tagged text available there will also be a much larger amount of untagged text available, a resource not utilized by purely supervised training algorithms.

One significant difference between this approach and that taken in using the Baum-Welch algorithm is that here the supervision influences the learner after unsupervised training, whereas when using tagged text to bias the initial probabilities for Baum-Welch training, supervision influences the learner prior to unsupervised training. The latter approach has the potential weakness of unsupervised training erasing what was learned from the manually annotated corpus. For example, in [Merialdo, 1995], extracting probability estimates from a 50,000 word manually tagged corpus gave a test set accuracy of 95.4%. After applying ten iterations of the Baum-Welch algorithm, accuracy dropped to 94.4%.

Using the transformations learned in the above unsupervised training experiment run on the Penn Treebank, we apply these transformations to a separate training corpus. New supervised transformations are then learned by comparing the tagged corpus that results from applying these transformations with the correct tagging, as indicated in the manually annotated training corpus.

In table 2, we show tagging accuracy on a separate test set using different sizes of manually annotated corpora. In each case, a 120,000 word untagged corpus was used for initial unsupervised training. This table also gives results from supervised training using the annotated corpus, without any prior unsupervised training.[12] In all cases, the combined training outperformed the purely supervised training at no added cost in terms of annotated

[12] The purely supervised learning algorithm is the same as that described in [Brill, 1994], except there the most likely tag for every word in the dictionary is provided to the learner.

Supervised Training Corpus Size (Words)	% Correct Using Unsupervised Transformations	% Correct Not Using Unsup. Transformations
0	95.1	90.8
400	95.4	91.8
1200	95.5	92.9
4000	95.7	93.9
7600	95.8	94.6
10300	96.0	95.1
22300	96.3	95.5
44400	96.6	96.1
61400	96.7	96.3
88200	96.8	96.5

Table 2: Unsupervised + Supervised vs. Purely Supervised Training.

training text.

Conclusions

In this paper, we have presented a new algorithm for unsupervised training of a rule-based part of speech tagger. The rule-based tagger trained using this algorithm significantly outperforms the traditional method of applying the Baum-Welch algorithm for unsupervised training of a stochastic tagger, and achieves comparable performance to a class-based Baum-Welch training algorithm. In addition, we have shown that by combining unsupervised and supervised learning, we can obtain a tagger that significantly outperforms a tagger trained using purely supervised learning. We are encouraged by these results, and expect an improvement in performance when the number of transformation templates provided to the unsupervised learner increases beyond the four currently used. We have also demonstrated that overtraining, a problem in Baum-Welch training, is not a problem in transformation-based learning.

References

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

[Black *et al.*, 1992] Black, E.; Jelinek, F.; Lafferty, J.; Mercer, R.; and Roukos, S. 1992. Decision tree models applied to the labeling of text with parts-of-speech. In *Darpa Workshop on Speech and Natural Language*. Harriman, N.Y.

[Brill and Resnik, 1994] Brill, E. and Resnik, P. 1994. A transformation-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING-1994)*, Kyoto, Japan.

[Brill, 1992] Brill, E. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy.

[Brill, 1993a] Brill, E. 1993a. Automatic grammar induction and parsing free text: A transformation-based approach. In *Proceedings of the 31st Meeting of the Association of Computational Linguistics*, Columbus, Oh.

[Brill, 1993b] Brill, E. 1993b. Transformation-based error-driven parsing. In *Proceedings of the Third International Workshop on Parsing Technologies*, Tilburg, The Netherlands.

[Brill, 1994] Brill, E. 1994. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Wa.

[Charniak *et al.*, 1993] Charniak, E.; Hendrickson, C.; Jacobson, N.; and Perkowitz, M. 1993. Equations for part of speech tagging. In *Proceedings of the Conference of the American Association for Artificial Intelligence (AAAI-93).*

[Church, 1988] Church, K. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing, ACL.*

[Cutting *et al.*, 1992] Cutting, D.; Kupiec, J.; Pedersen, J.; and Sibun, P. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy.

[DeMarcken, 1990] DeMarcken, C. 1990. Parsing the lob corpus. In *Proceedings of the 1990 Conference of the Association for Computational Linguistics.*

[Derose, 1988] Derose, S. 1988. Grammatical category disambiguation by statistical optimization. *Computational Linguistics* 14.

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

[Francis and Kucera, 1982] Francis, W. and Kucera, H. 1982. *Frequency analysis of English usage: Lexicon and grammar.* Houghton Mifflin, Boston.

[Harris, 1962] Harris, Z. 1962. *String Analysis of Language Structure.* Mouton and Co., The Hague.

[Hindle, 1989] Hindle, D. 1989. Acquiring disambiguation rules from text. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics.*

[Huang *et al.*, 1994] Huang, C.; Son-Bell, M.; and Baggett, D. 1994. Generation of pronunciations from orthographies using transformation-based error-driven learning. In *International Conference on Speech and Language Processing (ICSLP)*, Yokohama, Japan.

[Jelinek, 1985] Jelinek, F. 1985. *Self-Organized Language Modelling for Speech Recognition.* Dordrecht. In Impact of Processing Techniques on Communication, J. Skwirzinski, ed.

[Klein and Simmons, 1963] Klein, S. and Simmons, R. 1963. A computational approach to grammatical coding of English words. *JACM* 10.

[Kupiec, 1992] Kupiec, J. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer speech and language* 6.

[Lin *et al.*, 1994] Lin, Y.; Chiang, T.; and Su, K. 1994. Automatic model refinement with an application to tagging. In *Proceedings of the 15th International Conference on Computational Linguistics.*

[Marcus *et al.*, 1993] Marcus, M.; Santorini, B.; and Marcinkiewicz, M. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2).

[Merialdo, 1995] Merialdo, B. 1995. Tagging english text with a probabilistic model. *Computational Linguistics: To Appear.*

[Ramshaw and Marcus, 1994] Ramshaw, L. and Marcus, M. 1994. Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In *The Balancing Act: Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Language*, New Mexico State University.

[Roche and Schabes, 1995] Roche, E. and Schabes, Y. 1995. Deterministic part of speech tagging with finite state transducers. *Computational Linguistics: To Appear.*

[Schutze and Singer, 1994] Schutze, H. and Singer, Y. 1994. Part of speech tagging using a variable memory Markov model. In *Proceedings of the Association for Computational Linguistics.*

[Weischedel *et al.*, 1993] Weischedel, R.; Meteer, M.; Schwartz, R.; Ramshaw, L.; and Palmucci, J. 1993. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics.*