Classifier Combination for Improved Lexical Disambiguation

Eric Brill and Jun Wu
Department of Computer Science
Johns Hopkins University
Baltimore, Md. 21218 USA
{brill,junwu}@cs.jhu.edu

Abstract

One of the most exciting recent directions in machine learning is the discovery that the combination of multiple classifiers often results in significantly better performance than what can be achieved with a single classifier. In this paper, we first show that the errors made from three different state of the art part of speech taggers are strongly complementary. Next, we show how this complementary behavior can be used to our advantage. By using contextual cues to guide tagger combination, we are able to derive a new tagger that achieves performance significantly greater than any of the individual taggers.

Introduction

Part of speech tagging has been a central problem in natural language processing for many years. Since the advent of manually tagged corpora such as the Brown Corpus and the Penn Treebank (Francis(1982), Marcus(1993)), the efficacy of machine learning for training a tagger has been demonstrated using a wide array of techniques, including: Markov models, decision trees, connectionist machines, transformations, nearest-neighbor algorithms, and maximum entropy (Weischedel(1993), Black(1992), Schmid(1994), Brill(1995),Daelemans(1995),Ratnaparkhi(1996)). All of these methods seem to achieve roughly comparable accuracy.

The fact that most machine-learning-based taggers achieve comparable results could be attributed to a number of causes. It is possible that the 80/20 rule of engineering is applying: a certain number of tagging instances are relatively simple to disambiguate and are therefore being successfully tagged by all approaches, while another percentage is extremely difficult to disambiguate, requiring deep linguistic knowledge, thereby causing all taggers to err. Another possibility could be that all of the different machine learning techniques are essentially doing the same thing. We know that the features used by the different algorithms are very similar, typically the words and tags within a small window from the word being tagged. Therefore it could be possible that they all end up learning the same information, just in different forms.

In the field of machine learning, there have been many recent results demonstrating the efficacy of combining classifiers.[1] In this paper we explore whether classifier combination can result in an overall improvement in lexical disambiguation accuracy.

1 Different Tagging Algorithms

The experiments described in this paper are based on four popular tagging algorithms, all of which have readily available implementations. These taggers are described below.

1.1 Unigram Tagging

This is by far the simplest of tagging algorithms. Every word is simply assigned its most likely part of speech, regardless of the context in which it appears. Surprisingly, this simple tagging method achieves fairly high accuracy. Accuracies of 90-94% are typical. In the unigram tagger used in our experiments, for words that do not appear in the lexicon we use a

[1] See Dietterich(1997) for a good summary of these techniques.

collection of simple manually-derived heuristics to guess the proper tag for the word.

1.2 N-Gram Tagging

N-gram part of speech taggers (Bahl(1976), Church(1992), Weischedel(1993)) are perhaps the most widely used of tagging algorithms. The basic model is that given a word sequence W, we try to find the tag sequence T that maximizes P(T|W). This can be done using the Viterbi algorithm to find the T that maximizes: P(T)*P(W|T). In our experiments, we use a standard trigram tagger using deleted interpolation (Jelinek (1980)) and used suffix information for handling unseen words (as was done in Weischedel (1993)).

1.3 Transformation-Based Tagging

In transformation-based tagging (Brill (1995)), every word is first assigned an initial tag, This tag is the most likely tag for a word if the word is known and is guessed based upon properties of the word if the word is not known. Then a sequence of rules are applied that change the tags of words based upon the contexts they appear in. These rules are applied deterministically, in the order they appear in the list. As a simple example, if **race** appears in the corpus most frequently as a noun, it will initially be mistagged as a noun in the sentence :

We can **race** all day long.

The rule *Change a tag from NOUN to VERB if the previous tag is a MODAL* would be applied to the sentence, resulting in the correct tagging. The environments used for changing a tag are the words and tags within a window of three words. For our experiments, we used a publicly available implementation of transformation-based tagging,[2] retrained on our training set.

1.4 Maximum-Entropy Tagging

The maximum-entropy framework is a probabilistic framework where a model is found that is consistent with the observed data and is maximally agnostic with respect to all parameters for which no data exists. It is a nice framework for combining multiple constraints. Whereas the transformation-based tagger enforces multiple constraints by having multiple rules fire, the maximum-entropy tagger can have all of these constraints play a role at setting the probability estimates for the model's parameters. In Ratnaparkhi (1996), a maximum entropy tagger is presented. The tagger uses essentially the same parameters as the transformation-based tagger, but employs them in a different model. For our experiments, we used a publicly available implementation of maximum-entropy tagging,[3] retrained on our training set.

[2] http://www.cs.jhu.edu/~brill

2 Tagger Complementarity

All experiments presented in this paper were run on the Penn Treebank Wall Street Journal corpus (Marcus (1993)). The corpus was divided into approximately 80% training and 20% testing, giving us approximately 1.1 million words of training data and 265,000 words of test data. The test set was not used in any way in training, so the test set *does* contain unknown words.

In Figure 1 we show the relative accuracies of the four taggers. In parentheses we include tagger accuracy when only ambiguous and unknown words are considered.[4]

Tagger	Accuracy (%)	Num Errors
Unigram	93.26 (87.9)	17856
Trigram	96.36 (93.8)	9628
Transform.	96.61 (94.3)	8980
Max. Ent.	96.83 (94.7)	8400

Figure 1: Relative Tagger Accuracies

Next, we examine just how different the errors of the taggers are. We define the complementary rate of taggers A and B as :

[3] http://www.cis.upenn.edu/~adwait

[4] It is typical in tagging papers to give results in ambiguity resolution over all words, including words that are unambiguous. Correctly tagging words that only can have one label contributes to the accuracy. We see in Figure 1 that when accuracy is measured on truly ambiguous words, the numbers are lower. In this paper we stick to the convention of giving results for *all* words, including unambiguous ones.

$$Comp(A,B) = (1 - \frac{\#\text{ of common errors}}{\#\text{ of errors in A only}}) * 100$$

In other words, Comp(A,B) measures the percentage of time when tagger A is wrong that tagger B is correct. In Figure 2 we show the complementary rates between the different taggers. For instance, when the maximum entropy tagger is wrong, the transformation-based tagger is right 37.7% of the time, and when the transformation-based tagger is wrong, the maximum entropy tagger is right 41.7% of the time.

	Unigram	Trigram	Transf.	MaxEnt
Unigram	0	32.1	20.0	34.9
Trigram	63.4	0	34.6	33.5
Transf.	59.7	39.0	0	37.7
MaxEnt	69.4	42.0	41.7	0

Figure 2: Comp(A,B). Row = A, Column = B

The complementary rates are quite high, which is encouraging, since this sets the upper bound on how well we can do in combining the different classifiers. If all taggers made the same errors, or if the errors that lower-accuracy taggers made were merely a superset of higher-accuracy tagger errors, then combination would be futile.

In addition, a tagger is much more likely to have misclassified the tag for a word in instances where there is disagreement with at least one of the other classifiers than in the case where all classifiers agree. In Figure 3 we see, for instance that while the overall error rate for the Maximum Entropy tagger is 3.17%, in cases where there is disagreement between the four taggers the Maximum Entropy tagger error rate jumps to 27.1%. And discarding the unigram tagger, which is significantly less accurate than the others, when there is disagreement between the Maximum Entropy, Transformation-based and Trigram taggers, the Maximum Entropy tagger error rate jumps up to 43.7%. These cases account for 58% of the total errors the Maximum Entropy tagger makes (4833/8400).

Next, we check whether tagger complementarity is additive. In Figure 4, the first row shows the additive error rate an oracle could achieve on the test set if the oracle could pick between the different outputs of the taggers. For example, when the oracle can examine the output of the Maximum Entropy, Transformation-Based and Trigram taggers, it could achieve an error rate of 1.62%. The second row shows the additive error rate reduction the oracle could achieve. If the oracle is allowed to choose between all four taggers, a 55.5% error rate reduction is obtained over the Maximum Entropy tagger error rate. If the unigram output is discarded, the oracle improvement drops down to 48.8% over Maximum Entropy tagger error rate.

	Max.Ent	Trans-form	Tri-gram	Uni-gram
Overall Error Rate	3.17% (8400)	3.39 (8980)	3.64 (9628)	6.74 (17856)
Error Rate When Disagreement	27.1 (5535)	29.9 (6115)	33.1 (6763)	73.4 (14991)
Error Rate When Disagreement (excluding unigram)	43.7 (4833)	49.0 (5413)	54.9 (6061)	

Figure 3: Disagreement Is A Strong Indication of Error

	MaxEnt	+Transf.	+Tri-gram	+Uni-gram
% of time all are wrong	3.17	1.98	1.62	1.41
% Oracle Improvement		37.7	48.8	55.5

Figure 4 : Complementarity Is Additive.

From these results, we can conclude that there is at least hope that improvments can be gained by combining the output of different taggers. We can also conclude that the improvements we expect are somewhat additive, meaning the more taggers we combine, the better results we should expect.

3 Tagger Combination

The fact that the errors the taggers make are strongly complementary is very encouraging. If all taggers made the exact same errors, there would obviously be no chance of improving accuracy through classifier combination. However, note that the high complementary rate between tagger errors in itself does not necessarily imply that there is anything to be gained by classifier combination.

We ran experiments to determine whether the outputs of the different taggers could be effectively combined. We first explored combination via simple majority-wins voting. Next, we attempted to automatically acquire contextual cues that learned both which tagger to believe in which contexts and what tags are indicated by different patterns of tagger outputs. Both the word environments and the tagger outputs for the word being tagged and its neighbors are used as cues for predicting the proper tag.

3.1 Simple Voting

The simplest combination scheme is to have the classifiers vote. The part of speech that appeared as the choice of the largest number of classifiers is picked as the answer, with some method being specified for breaking ties. We tried simple voting, using the Maximum Entropy, Transformation-Based and Trigram taggers. In case of ties (all taggers disagree), the Maximum Entropy tagger output is chosen, since this tagger had the highest overall accuracy (this was determined by using a subset of the training set, not by using the test set). The results are shown in Figure 5. Simple voting gives a net reduction in error of 6.9% over the best of the three taggers. This difference is significant at a >99% confidence level.

Tagger	Error Rate	Num Errors
Max Ent	3.2%	8400
Simple Voting	3.0%	7823

Figure 5 Results of Simple Voting

3.2 Contextual Cues

Next, we try to exploit the idiosyncracies of the different taggers. Although the Maximum Entropy, Transformation-based and Trigram taggers use essentially the same types of contextual information for disambiguation, this information is exploited differently in each case. Our hope is that there is some regularity to these differences, which would then allow us to learn what conditions suggest that we should trust one tagger output over another.

We used a version of example-based learning to determine whether these tagger differences could be exploited.[5] To determine the tag of a word, we use the previous word, current word, next word, and the output of each tagger for the previous, current and next word. See Figure 6.

$Word_{j-1}$	**$Word_j$**	$Word_{j+1}$
$Unigram\_Tag_{j-1}$	$Unigram\_Tag_j$	$Unigram\_Tag_{j+1}$
$Trigram\_Tag_{j-1}$	$Trigram\_Tag_j$	$Trigram\_Tag_{j+1}$
$Transform\_Tag_{j-1}$	$Transform\_Tag_j$	$Transform\_Tag_{j+1}$
$MaxEnt\_Tag_{j-1}$	$MaxEnt\_Tag_j$	$MaxEnt\_Tag_{j+1}$

Figure 6 Features Used To Determine The Proper Tag for Word j.

For each such context in the training set, we store the probabilities of what correct tags appeared in that context. When the tag distribution for a context has low entropy, it is a very good predictor of the correct tag when the identical environment occurs in unseen data. The problem is that these environments are very specific, and will have low overall recall in a novel corpus. To account for this, we must back off to more general contexts when we encounter an environment in the test set that did not occur in the training set. This is done by specifying an order in which fields should be ignored until a match is found. The back-off ordering is learned automatically.

We ran two variants of this experiment. In the first case, given an instance in the test set, we find the most specific matching example in the training set, using the prespecified back-off ordering, and see what the most probable tag was in the training set for that environment. This is then chosen as the tag for the word. Note that this method is capable of learning to assign a tag that none of the taggers assigned. For instance, it could be the case that when the Unigram tagger thinks the tag should be X, and the Trigram and Maximum Entropy taggers think it should be Y, then the true tag is most frequently Z.

In the second experiment, we use contexts to specify which tagger to trust, rather than which tag to output. Again the most specific context is found, but here we check which tagger has the highest probability of being correct in this particular context. For instance, we may learn that the Trigram tagger is most accurate at tagging the word *up* or that the Unigram tagger does best at tagging the word

[5] Example-based learning has also been applied succesfully in building a single part of speech tagger (Daelemans(1996)).

race when the word that follows is *and*. The results are given in Figure 7. We see that while simple voting achieves an error reduction of 6.9%, using contexts to choose a tag gives an error reduction of 9.8% and using contexts to choose a tagger gives an error reduction of 10.4%.

Tagger	Error Rate	Num Errors
Max Ent	3.2%	8400
Simple Voting	3.0%	7823
Context: Pick Tag	2.9%	7580
Context: Pick Tagger	2.8%	7529

Figure 7 Error Rate Reduction For Different Tagger Combination Methods

Conclusion

In this paper, we showed that the error distributions for three popular state of the art part of speech taggers are highly complementary. Next, we described experiments that demonstrated that we can exploit this complementarity to build a tagger that attains significantly higher accuracy than any of the individual taggers.

In the future, we plan to expand our repertoire of base taggers, to determine whether performance continues to improve as we add additional systems. We also plan to explore different methods for combining classifier outputs. We suspect that the features we have chosen to use for combination are not the optimal set of features. We need to carefully study the different algorithms to find possible cues that can indicate where a particular tagger performs well. We hope that by following these general directions, we can further exploit differences in classifiers to improve accuracy in lexical disambiguation.

References

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.* Darpa Workshop on Speech and Natural Language, Harriman, N.Y.

Brill, E. (1995). *Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging.* Computational Linguistics.

Daelemans W. (1996). *MBT: A Memory-Based Part of Speech Tagger-Generator.* Proceedings of the Workshop on Very Large Corpora, Copenhagen

Dietterich T. (1997). *Machine-Learning Research: Four Current Directions.* AI Magazine. Winter 1997, pp97-136.

Francis W. and Kucera H. (1982) *Frequency analysis of English usage: Lexicon and grammar.* Houghton Mifflin.

Jelinek F and Mercer R (1980). *Interpolated Estimation of Markov Source Parameters from Sparse Data.* In Pattern Recognition in Practice, E. Gelsema and L. Kanal, Eds. Amsterdam: North-Holland.

Marcus M., Santorini B. and Marcinkiewicz M. (1993) *Building a large annotated corpus of English: the Penn Treebank.* Computational Linguistics.

Ratnaparkhi A. (1996*). A Maximum Entropy Part-of-Speech Tagger.* Proceedings of the First Empirical Methods in Natural Language Processing Conference. Philadelphia, Pa.

Schmid H. (1994). *Part of Speech Tagging With Neural Networks.* Proceedings of COLING, Yokohama, Japan.

Weischedel R., Meteer M., Schwartz R., Ramshaw L. and Palmucci, J. (1993). *Coping with ambiguity and unknown words through probabilistic models.* Computational Linguistics.