Some Advances in Transformation-Based Part of Speech Tagging*

Eric Brill
Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
brill@goldilocks.lcs.mit.edu

Abstract

Most recent research in trainable part of speech taggers has explored stochastic tagging. While these taggers obtain high accuracy, linguistic information is captured indirectly, typically in tens of thousands of lexical and contextual probabilities. In (Brill 1992), a trainable rule-based tagger was described that obtained performance comparable to that of stochastic taggers, but captured relevant linguistic information in a small number of simple non-stochastic rules. In this paper, we describe a number of extensions to this rule-based tagger. First, we describe a method for expressing lexical relations in tagging that stochastic taggers are currently unable to express. Next, we show a rule-based approach to tagging unknown words. Finally, we show how the tagger can be extended into a k-best tagger, where multiple tags can be assigned to words in some cases of uncertainty.

Introduction

When automated part of speech tagging was initially explored (Klein & Simmons 1963; Harris 1962), people manually engineered rules for tagging, sometimes with the aid of a corpus. As large corpora became available, it became clear that simple Markov-model based stochastic taggers that were automatically trained could achieve high rates of tagging accuracy (Jelinek 1985). Markov-model based taggers assign a sentence the tag sequence that maximizes $Prob(\text{word}|\text{tag}) * Prob(\text{tag}|\text{previous n tags})$. These probabilities can be estimated directly from a manually tagged corpus.[1] Stochastic taggers have a number of advantages over the manually built taggers, including obviating the need for laborious manual rule construction, and possibly capturing useful information that may not have been noticed by the human engineer. However, stochastic taggers have the disadvantage that linguistic information is only captured indirectly, in large tables of statistics. Almost all recent work in developing automatically trained part of speech taggers has been on further exploring Markov-model based tagging (Jelinek 1985; Church 1988; Derose 1988; DeMarcken 1990; Merialdo 1991; Cutting *et al.* 1992; Kupiec 1992; Charniak *et al.* 1993; Weischedel *et al.* 1993).

In (Brill 1992), a trainable rule-based tagger is described that achieves performance comparable to that of stochastic taggers. Training this tagger is fully automated, but unlike trainable stochastic taggers, linguistic information is encoded directly in a set of simple non-stochastic rules. In this paper, we describe some extensions to this rule-based tagger. These include a rule-based approach to: lexicalizing the tagger, tagging unknown words, and assigning the k-best tags to a word. All of these extensions, as well as the original tagger, are based upon a learning paradigm called transformation-based error-driven learning. This learning paradigm has shown promise in a number of other areas of natural language processing, and we hope that the extensions to transformation-based learning described in this paper can carry over to other domains of application as well.[2]

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, and syntactic parsing (Brill 1992; 1993a; 1993b). A similar approach is being explored for machine translation (Su, Wu, & Chang

*This paper appears in the proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94). This research was supported by ARPA under contract N00014-89-J-1332, monitored through the Office of Naval Research.

[1]One can also estimate these probabilities without a manually tagged corpus, using a hidden Markov model. However, it appears to be the case that directly estimating probabilities from even a very small manually tagged corpus gives better results than training a hidden Markov model on a large untagged corpus (see (Merialdo 1991)).

[2]The programs described in this paper can be obtained by contacting the author.

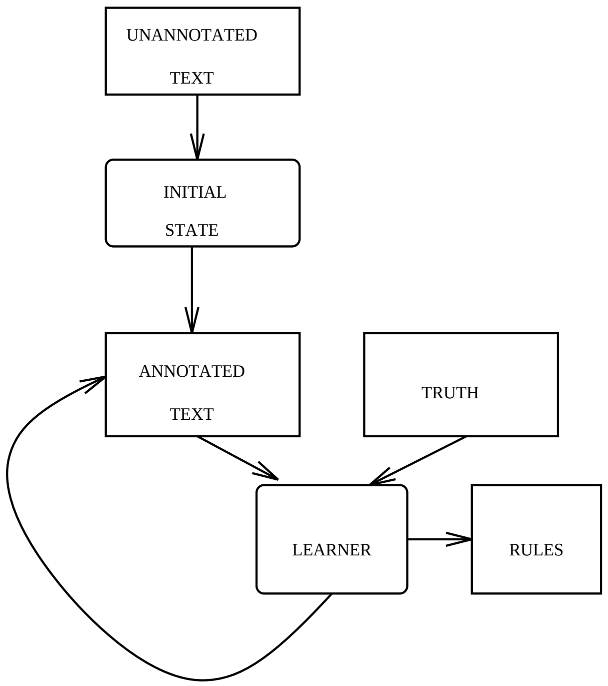

Figure 1: Transformation-Based Error-Driven Learning.

1992). Figure 1 illustrates the learning process. First, unannotated text is passed through the 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*,[3] 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 described in this paper, 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, and (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.

[3]As specified in a manually annotated corpus.

An Earlier Tranformation-Based Tagger

The original transformation-based tagger (Brill 1992) works as follows. The initial state annotator assigns each word its most likely tag as indicated in the training corpus. The most likely tag for unknown words is guessed based on a number of features, such as whether the word is capitalized, and what the last three letters of the word are. The allowable transformation templates are:

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

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

where a,b,z and w are variables over the set of parts of speech. To learn a transformation, the learner in essence applies every possible transformation,[4] counts the number of tagging errors after that transformation is applied, and chooses that transformation resulting in the greatest error reduction.[5] Learning stops when no transformations can be found whose application reduces errors beyond some prespecified threshold. An example of a transformation that was learned is: change the tagging of a word from **noun** to **verb** if the previous word is tagged as a **modal**. Once the system is trained, a new sentence is tagged by applying the initial state annotator and then applying each transformation, in turn, to the sentence.

Lexicalizing the Tagger

No relationships between words are directly captured in stochastic taggers. In the Markov model, state transition probabilities ($P(Tag_i|Tag_{i-1} \ldots Tag_{i-n})$) express the likelihood of a tag immediately following n other tags, and emit probabilities ($P(Word_j|Tag_i)$) express the likelihood of a word given a tag. Many useful relationships, such as that between a word and the previous word, or between a tag and the following word, are not directly captured by Markov-model based taggers. The same is true of the earlier transformation-based tagger, where transformation templates did not make reference to words.

To remedy this problem, the transformation-based tagger was extended by adding contextual transformations that could make reference to words as well as

[4]All possible instantiations of transformation templates.

[5]The search is data-driven, so only a very small percentage of possible transformations really need be examined.

part of speech tags. The transformation templates that were added are:

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

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

where w and x are variables over all words in the training corpus, and z is a variable over all parts of speech.

Below we list two lexicalized transformations that were learned:[6]

Change the tag:

(12) From **preposition** to **adverb** if the word two positions to the right is **as**.
(16) From **non-3rd person singular present verb** to **base form verb** if one of the previous two words is **n't**.[7]

The Penn Treebank tagging style manual specifies that in the collocation *as ... as*, the first *as* is tagged as an adverb and the second is tagged as a preposition. Since *as* is most frequently tagged as a preposition in the training corpus, the initial state tagger will mistag the phrase *as tall as* as:

as/**preposition** tall/adjective as/preposition

The first lexicalized transformation corrects this mistagging. Note that a stochastic tagger trained on our training set would not correctly tag the first occurrence of *as*. Although adverbs are more likely than prepositions to follow some verb form tags, the fact that $P(as|preposition)$ is much greater than $P(as|adverb)$, and $P(adjective|preposition)$ is much greater than $P(adjective|adverb)$ lead to *as* being incorrectly tagged as a preposition by a stochastic tagger. A trigram tagger will correctly tag this collocation in some instances, due to the fact that $P(preposition|\text{adverb adjective})$ is greater than $P(preposition|\text{preposition adjective})$, but the outcome will be highly dependent upon the context in which this collocation appears.

The second transformation arises from the fact that when a verb appears in a context such as *We do n't* ___ or *We did n't usually* ___, the verb is in base form. A stochastic trigram tagger would have to capture this linguistic information indirectly from frequency counts of all trigrams of the form:[8]

*	ADVERB	PRESENT_VERB
*	ADVERB	BASE_VERB
ADVERB	*	PRESENT_VERB
ADVERB	*	BASE_VERB

and from the fact that $P(n't|ADVERB)$ is fairly high.

In (Weischedel *et al.* 1993), results are given when training and testing a Markov-model based tagger on the Penn Treebank Tagged Wall Street Journal Corpus. They cite results making the closed vocabulary assumption that all possible tags for all words in the test set are known. When training contextual probabilities on 1 million words, an accuracy of 96.7% was achieved. Accuracy dropped to 96.3% when contextual probabilities were trained on 64,000 words. We trained the transformation-based tagger on 600,000 words from the same corpus, making the same closed vocabulary assumption,[9] and achieved an accuracy of 97.2% on a separate 150,000 word test set. The transformation-based learner achieved better performance, despite the fact that contextual information was captured in only 267 simple nonstochastic rules, as opposed to 10,000 contextual probabilities that were learned by the stochastic tagger. To see whether lexicalized transformations were contributing to the accuracy rate, we ran the exact same test using the tagger trained using the earlier transformation template set, which contained no transformations making reference to words. Accuracy of that tagger was 96.9%. Disallowing lexicalized transformations resulted in an 11% increase in the error rate. These results are summarized in table 1.

When transformations are allowed to make reference to words and word pairs, some relevant information is probably missed due to sparse data. We are currently exploring the possibility of incorporating word classes into the rule-based learner in hopes of overcoming this problem. The idea is quite simple. Given a source of word class information, such as WordNet (Miller 1990), the learner is extended such that a rule is allowed to make reference to parts of speech, words, and word

[6] All experiments were run on the Penn Treebank tagged Wall Street Journal corpus, version 0.5 (Marcus, Santorini, & Marcinkiewicz 1993).

[7] In the Penn Treebank, *n't* is treated as a separate token, so *don't* becomes *do/VB-NON3rd-SING n't/ADVERB*.

[8] Where a star can match any part of speech tag.

[9] In both (Weischedel *et al.* 1993) and here, the test set was incorporated into the lexicon, but was not used in learning contextual information. Testing with no unknown words might seem like an unrealistic test. We have done so for three reasons (We show results when unknown words are included later in the paper): (1) to allow for a comparison with previously quoted results, (2) to isolate known word accuracy from unknown word accuracy, and (3) in some systems, such as a closed vocabulary speech recognition system, the assumption that all words are known is valid.

Method	Training Corpus Size (Words)	# of Rules or Context. Probs.	Acc. (%)
Stochastic	64 K	6,170	96.3
Stochastic	1 Million	10,000	96.7
Rule-Based w/o Lex. Rules	600 K	219	96.9
Rule-Based With Lex. Rules	600 K	267	97.2

Table 1: Comparison of Tagging Accuracy With No Unknown Words

classes, allowing for rules such as *Change the tag from X to Y if the following word belongs to word class Z.* This approach has already been successfully applied to a system for prepositional phrase disambiguation (Brill 1993b).

Unknown Words

In addition to not being lexicalized, another problem with the original transformation-based tagger was its relatively low accuracy at tagging unknown words.[10] In the initial state annotator for tagging, words are assigned their most likely tag, estimated from a training corpus. In the original formulation of the rule-based tagger, a rather ad-hoc algorithm was used to guess the most likely tag for words not appearing in the training corpus. To try to improve upon unknown word tagging accuracy, we built a transformation-based learner to learn rules for more accurately guessing the most likely tag for words not seen in the training corpus. If the most likely tag for unknown words can be assigned with high accuracy, then the contextual rules can be used to improve accuracy, as described above.

In the transformation-based unknown-word tagger, the initial state annotator naively labels the most likely tag for unknown words as proper noun if capitalized and common noun otherwise.[11]

Below we list the set of allowable transformations:

Change the tag of an unknown word (from X) to Y if:

1. Deleting the prefix x, $|x| <= 4$, results in a word (x is any string of length 1 to 4).
2. The first (1,2,3,4) characters of the word are x.
3. Deleting the suffix x, $|x| <= 4$, results in a word.
4. The last (1,2,3,4) characters of the word are x.
5. Adding the character string x as a suffix results in a word ($|x| <= 4$).
6. Adding the character string x as a prefix results in a word ($|x| <= 4$).
7. Word W ever appears immediately to the left (right) of the word.
8. Character Z appears in the word.

An unannotated text can be used to check the conditions in all of the above transformation templates. Annotated text is necessary in training to measure the effect of transformations on tagging accuracy. Below are the first 10 transformation learned for tagging unknown words in the Wall Street Journal corpus:

Change tag:

1. From **common noun** to **plural common noun** if the word has suffix **-s**[12]
2. From **common noun** to **number** if the word has character **.**
3. From **common noun** to **adjective** if the word has character **-**
4. From **common noun** to **past participle verb** if the word has suffix **-ed**
5. From **common noun** to **gerund or present participle verb** if the word has suffix **-ing**
6. To **adjective** if adding the suffix **-ly** results in a word
7. To **adverb** if the word has suffix **-ly**
8. From **common noun** to **number** if the word **$** ever appears immediately to the left
9. From **common noun** to **adjective** if the word has suffix **-al**
10. From **noun** to **base form verb** if the word **would** ever appears immediately to the left.

Keep in mind that no specific affixes are prespecified. A transformation can make reference to any string of characters up to a bounded length. So while the first rule specifies the English suffix "s", the rule learner also considered such nonsensical rules as: *change a tag to adjective if the word has suffix "xhqr"*. Also, absolutely no English-specific information need be prespecified in the learner.[13]

We then ran the following experiment using 1.1 million words of the Penn Treebank Tagged Wall Street Journal Corpus. The first 950,000 words were used for

[10] This section describes work done in part while the author was at the University of Pennsylvania.

[11] If we change the tagger to tag all unknown words as common nouns, then a number of rules are learned of the form: **change tag to proper noun if the prefix is "E"**, since the learner is not provided with the concept of upper case in its set of transformation templates.

[12] Note that this transformation will result in the mistagging of *actress*. The 17th learned rule fixes this problem. This rule states: change a tag from **plural common noun** to **singular common noun** if the word has suffix ss.

[13] This learner has also been applied to tagging Old English. See (Brill 1993b).

training and the next 150,000 words were used for testing. Annotations of the test corpus were not used in any way to train the system. From the 950,000 word training corpus, 350,000 words were used to learn rules for tagging unknown words, and 600,000 words were used to learn contextual rules. 148 rules were learned for tagging unknown words, and 267 contextual tagging rules were learned. Unknown word accuracy on the test corpus was 85.0%, and overall tagging accuracy on the test corpus was 96.5%. To our knowledge, this is the highest overall tagging accuracy ever quoted on the Penn Treebank Corpus when making the open vocabulary assumption.

In (Weischedel *et al.* 1993), a statistical approach to tagging unknown words is shown. In this approach, a number of suffixes and important features are prespecified. Then, for unknown words:

$$p(W|T) = p(\text{unknown word}|T) * p(\text{Capitalize-feature}|T) * p(suffixes, hyphenation|T)$$

Using this equation for unknown word emit probabilities within the stochastic tagger, an accuracy of 85% was obtained on the Wall Street Journal corpus. This portion of the stochastic model has over 1,000 parameters, with 10^8 possible unique emit probabilities, as opposed to only 148 simple rules that are learned and used in the rule-based approach. We have obtained comparable performance on unknown words, while capturing the information in a much more concise and perspicuous manner, and without prespecifying any language-specific or corpus-specific information.

K-Best Tags

There are certain circumstances where one is willing to relax the one tag per word requirement in order to increase the probability that the correct tag will be assigned to each word. In (DeMarcken 1990; Weischedel *et al.* 1993), k-best tags are assigned within a stochastic tagger by returning all tags within some threshold of probability of being correct for a particular word.

We can modify the transformation-based tagger to return multiple tags for a word by making a simple modification to the contextual transformations described above. The initial-state annotator is the tagging output of the transformation-based tagger described above. The allowable transformation templates are the same as the contextual transformation templates listed above, but with the action *change tag X to tag Y* modified to *add tag X to tag Y* or *add tag X to word W*. Instead of changing the tagging of a word, transformations now add alternative taggings to a word.

When allowing more than one tag per word, there is a trade-off between accuracy and the average number of tags for each word. Ideally, we would like to achieve as large an increase in accuracy with as few extra tags as possible. Therefore, in training we find transformations that maximize precisely this function.

# of Rules	Accuracy	Avg. # of tags per word
0	96.5	1.00
50	96.9	1.02
100	97.4	1.04
150	97.9	1.10
200	98.4	1.19
250	99.1	1.50

Table 2: Results from k-best tagging.

In table 2 we present results from first using the one-tag-per-word transformation-based tagger described in the previous section and then applying the k-best tag transformations. These transformations were learned from a separate 240,000 word corpus. As a baseline, we did k-best tagging of a test corpus as follows. Each known word in the test corpus was tagged with all tags seen with that word in the training corpus and the five most likely unknown word tags were assigned to all words not seen in the training corpus.[14] This resulted in an accuracy of 99.0%, with an average of 2.28 tags per word. The rule-based tagger obtained the same accuracy with 1.43 tags per word, one third the number of additional tags as the baseline tagger.[15]

Conclusions

In this paper, we have described a number of extensions to previous work in rule-based part of speech tagging, including the ability to make use of lexical relationships previously unused in tagging, a new method for tagging unknown words, and a way to increase accuracy by returning more than one tag per word in some instances. We have demonstrated that the rule-based approach obtains competitive performance with stochastic taggers on tagging both unknown and known words. The rule-based tagger captures linguistic information in a small number of simple non-stochastic rules, as opposed to large numbers of lexical and contextual probabilities. Recently, we have begun to explore the possibility of extending these techniques to other problems, including learning pronunciation networks for speech recognition and learning mappings between sentences and semantic representations.

[14] Thanks to Fred Jelinek and Fernando Pereira for suggesting this baseline experiment.

[15] Unfortunately, it is difficult to find results to compare these k-best tag results to. In (DeMarcken 1990), the test set is included in the training set, and so it is difficult to know how this system would do on fresh text. In (Weischedel *et al.* 1993), a k-best tag experiment was run on the Wall Street Journal corpus. They quote the average number of tags per word for various threshold settings, but do not provide accuracy results.

References

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

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.*

Brill, E. 1993b. *A Corpus-Based Approach to Language Learning.* Ph.D. Dissertation, Department of Computer and Information Science, University of Pennsylvania.

Charniak, E.; Hendrickson, C.; Jacobson, N.; and Perkowitz, M. 1993. Equations for part of speech tagging. In *Proceedings of the Conferenceof the American Association for Artificial Intelligence.*

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, 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.*

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

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

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

Jelinek, F. 1985. *Markov Source modeling of text generation.* Dordrecht. In Impact of Processing Techniques on Communication, J. Skwirzinski, ed.

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

Kupiec, J. 1992. Robust part of speech tagging using a hidden markov model. *Computer Speech and Language.*

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

Merialdo, B. 1991. Tagging text with a probabilistic model. In *IEEE International Conference on Acoustics, Speech and Signal Processing.*

Miller, G. 1990. Wordnet: an on-line lexical database. *International Journal of Lexicography.*

Su, K.; Wu, M.; and Chang, J. 1992. A new quantitiative quality measure for machine translation. In *Proceedings of COLING-1992.*

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