Mistake-Driven Mixture of Hierarchical Tag Context Trees

Masahiko Haruno
NTT Communication Science Laboratories
1-1 Hikari-No-Oka Yokosuka-Shi
Kanagawa 239, Japan
haruno@cslab.kecl.ntt.co.jp

Yuji Matsumoto
NAIST
8916-5 Takayama-cho Ikoma-Shi
Nara 630-01, Japan
matsu@is.aist-nara.ac.jp

Abstract

This paper proposes a *mistake-driven mixture* method for learning a tag model. The method iteratively performs two procedures: *1.* constructing a tag model based on the current data distribution and *2.* updating the distribution by focusing on data that are not well predicted by the constructed model. The final tag model is constructed by mixing all the models according to their performance. To well reflect the data distribution, we represent each tag model as a *hierarchical tag* (i.e.,NTT[1] < proper noun < noun) *context tree*. By using the hierarchical tag context tree, the constituents of sequential tag models gradually change from broad coverage tags (e.g.,noun) to specific exceptional words that cannot be captured by general tags. In other words, the method incorporates not only frequent connections but also infrequent ones that are often considered to be collocational. We evaluate several tag models by implementing Japanese part-of-speech taggers that share all other conditions (i.e.,dictionary and word model) other than their tag models. The experimental results show the proposed method significantly outperforms both hand-crafted and conventional statistical methods.

1 Introduction

The last few years have seen the great success of stochastic part-of-speech (POS) taggers (Church, 1988; Kupiec, 1992; Charniak et al., 1993; Brill, 1992; Nagata, 1994). The stochastic approach generally attains 94 to 96% accuracy and replaces the labor-intensive compilation of linguistics rules by using an automated learning algorithm. However, practical systems require more accuracy because POS tagging is an inevitable pre-processing step for all practical systems.

To derive a new stochastic tagger, we have two options since stochastic taggers generally comprise two components: *word model* and *tag model*. The word model is a set of probabilities that a word occurs with a tag (part-of-speech) when given the preceding words and their tags in a sentence. On the contrary, the tag model is a set of probabilities that a tag appears after the preceding words and their tags.

The first option is to construct more sophisticated word models. (Charniak et al., 1993) reports that their model considers the roots and suffixes of words to greatly improve tagging accuracy for English corpora. However, the word model approach has the following shortcomings:

- For agglutinative languages such as Japanese and Chinese, the simple Bayes transfer rule is inapplicable because the word length of a sentence is not fixed in all possible segmentations[2]. We can only use simpler word models in these languages.
- Sophisticated word models largely depend on the target language. It is time-consuming to compile fine-grained word models for each language.

The second option is to devise a new tag model. (Schütze and Singer, 1994) have introduced a variable-memory-length tag model. Unlike conventional bi-gram and tri-gram models, the method selects the optimal length by using the context tree (Rissanen, 1983) which was originally introduced for use in data compression (Cover and Thomas, 1991). Although the variable-memory length approach remarkably reduces the number of parameters, tagging accuracy is only as good as conventional methods. Why didn't the method have higher accuracy ? The crucial problem for current

[1] NTT is an abbreviation of Nippon Telegraph and Telephone Corporation.

[2] In $P(w_i|t_i) = \frac{P(w_i)P(t_i|w_i)}{P(t_i)}$, $P(w_i)$ cannot be considered to be identical for all segmentations.

tag models is the set of collocational sequences of words that cannot be captured by just their tags. Because the maximal likelihood estimator (MLE) emphasizes the most frequent connections, an exceptional connection is placed in the same class as a frequent connection.

To tackle this problem, we introduce a new tag model based on the *mistake-driven mixture* of hierarchical tag context trees. Compared to Schütze and Singer's context tree (Schütze and Singer, 1994), the hierarchical tag context tree is extended in that the context is represented by a hierarchical tag set (i.e.,NTT < proper noun < noun). This is extremely useful in capturing exceptional connections that can be detected only at the word level.

To make the best use of the hierarchical context tree, the *mistake-driven mixture* method imitates the process in which linguists incorporate exceptional connections into hand-crafted rules: They first construct coarse rules which seems to cover broad range of data. They then try to analyze data by using the rules and extract exceptions that the rules cannot handle. Next they generalize the exceptions and refine the previous rules. The following two steps abstract the human algorithm for incorporating exceptional connections.

1. construct temporary rules which seem to well generalize given data.
2. try to analyze data by using the constructed rules and extract the exceptions that cannot be correctly handled, then return to the first step and focus on the exceptions.

To put the above idea into our learning algorithm, The *mistake-driven mixture* method attaches a weight vector to each example and iteratively performs the following two procedures in the training phase:

1. constructing a context tree based on the current data distribution (weight vector)
2. updating the distribution (weight vector) by focusing on data not well predicted by the constructed tree. More precisely, the algorithm reduces the weight of examples that are correctly handled.

For the prediction phase, it then outputs a final tag model by mixing all the constructed models according to their performance. By using the hierarchical tag context tree, the constituents of a series of tag models gradually change from broad coverage tags (e.g.,noun) to specific exceptional words that cannot be captured by general tags. In other words, the method incorporates not only frequent connections but also infrequent ones that are often considered to be exceptional.

The construction of the paper is as follows. Section 2 describes the stochastic POS tagging scheme and hierarchical tag setting. Section 3 presents a new probability estimator that uses a hierarchical tag context tree and Section 4 explains the mistake-driven mixture method. Section 5 reports a preliminary evaluation using Japanese newspaper articles. We tested several tag models by keeping all other conditions (i.e., dictionary and word model) identical. The experimental results show that the proposed method significantly outperforms both hand-crafted and conventional statistical methods. Section 6 concerns related works and Sections 7 concludes the paper.

2 Preliminaries

2.1 Basic Equation

In this section, we will briefly review the basic equations for part-of-speech tagging and introduce hierarchical-tag setting.

The tagging problem is formally defined as finding a sequence of tags $t_{1,n}$ that maximize the probability of input string L.

$$argmax_{t_{1,n}} P(w_{1,n}, t_{1,n}|L) = argmax_{t_{1,n}} \frac{P(w_{1,n}, t_{1,n}, L)}{P(L)}$$
$$\Leftrightarrow argmax_{t_{1,n}, w_{1,n} \in L} P(t_{1,n}, w_{1,n})$$

We break out $P(t_{1,n}, w_{1,n})$ as a sequence of the products of tag probability and word probability.

$$P(t_{1,n}, w_{1,n}) = \prod_{i=1}^{n} P(w_i|t_{1,i-1}, w_{1,i-1}) P(t_i|t_{1,i-1}, w_{1,i})$$

By approximating word probability as constrained only by its tag, we obtain equation (1). Equation (1) yields various types of stochastic taggers. For example, bi-gram and tri-gram models approximate their tag probability as $P(t_i|t_{i-1})$ and $P(t_i|t_{i-1}, t_{i-2})$, respectively. In the rest of the paper, we assume all tagging methods share the word model $P(w_i|t_i)$ and differ only in the tag model $P(t_i|t_{1,i-1}, w_{1,i})$.

$$argmax_{t_{1,n}, w_{1,n} \in L} \prod_{i=1}^{n} P(t_i|t_{1,i-1}, w_{1,i}) P(w_i|t_i) \quad (1)$$

2.2 Hierarchical Tag Set

To construct a tag model that captures exceptional connections, we have to consider word-level context as well as tag-level. In a more general form, we introduce a tag set that has a hierarchical structure. Our tag set has a three-level structure as shown in Figure 1. The topmost and the second level of the hierarchy are *part-of-speech* level and part-of-speech *subdivision* level respectively. Although stochastic taggers usually make use of *subdivision* level, *part-of-speech* level is remarkably robust

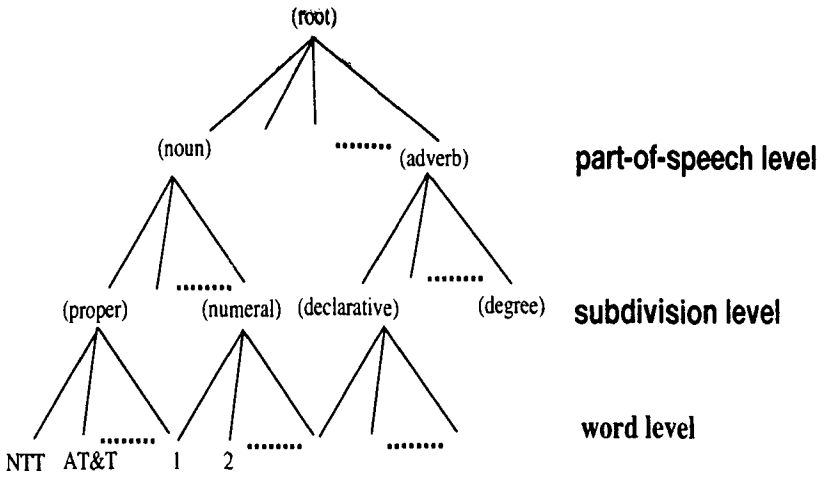

Figure 1: Hierarchical Tag Set

against data sparseness. The bottom level is *word* level and is indispensable in coping with exceptional and collocational sequences of words. Our objective is to construct a tag model that precisely evaluates $P(t_i|t_{1,i-1}, w_{1,i})$ (in equation (1)) by using the three-level tag set.

To construct this model, we have to answer the following questions.

1. Which level is appropriate for t_i ?
2. Which length is to be considered for $t_{1,i-1}$ and $w_{1,i}$?
3. Which level is appropriate for $t_{1,i-1}$ and $w_{1,i}$?

To resolve the first question, we fix t_i at *subdivision* level as is done in other tag models. The second and third questions are resolved by introducing *hierarchical tag context trees* and *mistake-driven mixture* method that are respectively described in Section 3 and 4.

Before moving to the next section, let us define the *basic tag set*. If all words are considered context candidates, the search space will be enormous. Thus, it is reasonable for the tagger to constrain the candidates to frequent open class words and closed class words. The *basic tag set* is a set of the most detailed context elements that comprises the *words* selected above and part-of-speech *subdivision* level.

3 Hierarchical Tag Context Tree

A hierarchical tag context tree is constructed by a two-step methodology. The first step produces a context tree by using the basic tag set. The second step then produces the hierarchical tag context tree. It generalizes the basic tag context tree and avoids over-fitting the data by replacing excessively specific context in the tree with more general tags. Finally, the generated tree is transformed into a finite automaton to improve tagging efficiency (Ron et al., 1997).

3.1 Constructing a Basic Tag Context Tree

In this section, we construct a basic tag context tree. Before going into detail of the algorithm, we briefly explain the context tree by using a simple binary case. The context tree was originally introduced in the field of data compression (Rissanen, 1983; Willems et al., 1995; Cover and Thomas, 1991) to represent how many times and in what context each symbol appeared in a sequence of symbols. Figure 2 exemplifies two context trees comprising binary symbols *'a'* and *'b'*. T(4) is constructed from the sequence *'baab'* and T(6) from *'baabab'*. The root node of T(4) explains that both *'a'* and *'b'* appeared twice in *'baab'* when no consideration is taken of previous symbols. The nodes of depth 1 represent an order 1 (bi-gram) model. The left node of T(4) represents that both *'a'* and *'b'* appeared only once after symbol *'a'*, while the right node of T(4) represents only *'a'* occurred once after *'b'*. In the same way, the node of depth 2 in T(6) represents an order 2 (tri-gram) context model.

It is straightforward to extend this binary tree to a basic tag context tree. In this case, context symbols *'a'* and *'b'* are replaced by an element of the basic tag set and the frequency table of each node then consists of the part-of-speech *subdivision* set.

The procedure *construct-btree* which constructs a basic tag context tree is given below. Let a set of *subdivision* tags to be $s_1,\cdots,s_n$. Let *weight*[*t*] be a weight vector attached to the *t*th example *x(t)*. Initial values of *weight*[*t*] are set to 1.

1. the only node, the root, is marked with the count table $(c(s_1,\lambda),\cdots, c(s_n,\lambda) = (0,\cdots,0))$.
2. Apply the following recursively. Let T(t-1) be

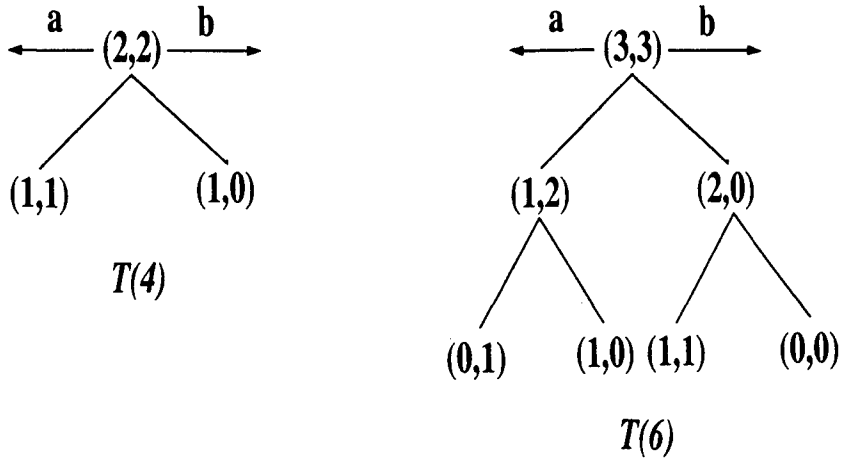

Figure 2: Context Trees for *'baab'* and *'baabab'*

the last constructed tree with counts of nodes z, (c(s_1,z),$\cdots$, c(s_n,z)). After the next symbol whose *subdivision* is *x(t)* is observed, generate the next tree T(t) as follows: follow the T(t-1), starting at the root and taking the branch indicated by each successive symbol in the past sequence by using basic tag level. For each node z visited, increment the component count c($x(t)$,z) by *weight*[t]. Continue until node w is a leaf node.

3. If w is a leaf, extend the tree by creating new leaves: c($x(t)$,ws_1)=$\cdots$=c($x(t)$,ws_n) = *weight*[t], c($\overline{x(t)}$,ws_1)=$\cdots$= c($\overline{x(t)}$,ws_n)=0. Define the resulting tree to be T(t).

3.2 Constructing a Hierarchical Tag Context Tree

This section delineates how a hierarchical tag context tree is constructed from a basic tag context tree. Before describing the algorithm, we prepare some definitions and notations.

Let A be a part-of-speech *subdivision* set. As described in the previous section, frequency tables of each node consist of the set A. At any node s of a context tree, let $n(a|s)$ and $\hat{P}(a|s)$ be the count of element a and its probability, respectively.

$$\hat{P}(a|s) = \frac{n(a|s)}{\sum_{b \subseteq A} n(b|s)}$$

We introduce an information-theoretical criteria $\Delta(sb)$ (Weinberger et al., 1995) to evaluate the gain of expanding a node s by its daughter sb.

$$\Delta(sb) = \sum_{a \subseteq A} n(a|sb) log \frac{\hat{P}(a|sb)}{\hat{P}(a|s)} \qquad (2)$$

$\Delta(sb)$ is the difference in optimal code lengths when symbols at node sb are compressed by using probability distribution $\hat{P}(\cdot|s)$ at node s and $\hat{P}(\cdot|sb)$ at node sb. Thus, the larger $\Delta(sb)$ is, the more meaningful it is to expand a node by sb.

Now, we go back to the hierarchical tag context tree construction. As illustrated in Figure 3, the generation process amounts to the iterative selection of b out of *word level*, *subdivision*, *part-of-speech* and *null* (no expansion). Let us look at the procedure from the information-theoretical viewpoint. Breaking out equation (2) as (3), $\Delta(sb)$ is represented as the product of the frequencies of all subdivision symbols at node sb and Kullback-Leibler (KL) divergence.

$$\Delta(sb) = n(sb) \sum_{a \subseteq A} \frac{n(a|sb)}{n(sb)} log \frac{\hat{P}(a|sb)}{\hat{P}(a|s)}$$

$$= n(sb) \sum_{a \subseteq A} \hat{P}(a|sb) log \frac{\hat{P}(a|sb)}{\hat{P}(a|s)}$$

$$= n(sb) D_{KL}(\hat{P}(\cdot|sb), \hat{P}(\cdot|s)) \qquad (3)$$

Because the KL divergence defines a distance measure between probability distributions, $\hat{P}(\cdot|sb)$ and $\hat{P}(\cdot|s)$, there is the following trade-off between the two terms of equation (3).

- The more general b is, the more *subdivision* symbols appear at node sb.
- The more specific b is, the more $\hat{P}(\cdot|s)$ and $\hat{P}(\cdot|sb)$ differ.

By using the trade-off, the optimal level of b is selected.

Table 1 summarizes the algorithm *construct-htree* that constructs the hierarchical tag context tree. First, *construct-htree* generates a basic tag context tree by calling *construct-btree*. Assume that the

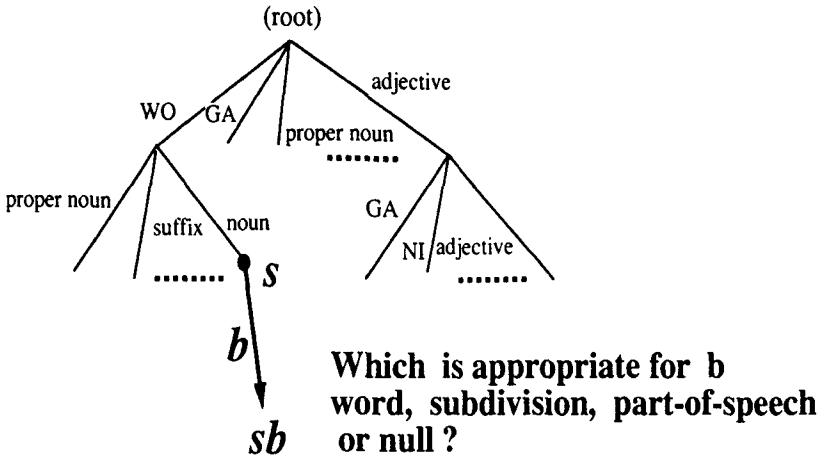

Figure 3: Constructing Hierarchical Tag Context Tree

training examples consist of a sequence of triples, $< p_t, s_t, w_t >$, in which p_t, s_t and w_t represent part-of-speech, subdivision and word, respectively. Eachtime the algorithm reads an example, it first reaches current leaf node s by following the past sequence, computes $\Delta(sb)$, and then selects the optimal b. The initially constructed basic tag context tree is used to compute $\Delta(sb)$s.

4 Mistake-Driven Mixture of Hierarchical Tag Context Trees

Up to this section, we introduced a new tag model that uses a single hierarchical tag context tree to cope with the exceptional connections that cannot be captured by just part-of-speech level. However, this approach has a clear limitation; the exceptional connections that do not occur so often cannot be detected by the single tree model. In such a case, the first term $n(sb)$ in equation (3) is enormous for general b and the tree is expanded by using more general symbols.

To overcome this limitation, we devised the *mistake-driven mixture* algorithm summarized in Table 4 which constructs T context trees and outputs the final tag model.

mistake-driven mixture sets the weights to 1 for all examples and repeats the following procedures T times. The algorithm first construct a hierarchical context tree by using the current weight vector. Example data are then tagged by the tree and the weights of correctly handled examples are reduced by equation (4). Finally, the final tag model is constructed by mixing T trees according to equation (5).

By using the *mistake-driven mixture* method, the constituents of a series of hierarchical tag context trees gradually change from broad coverage tags (e.g.,noun) to specific exceptional words that cannot be captured by *part-of-speech* and *subdivisions*. The method, by mixing different levels of trees, incorporates not only frequent connections but also infrequent ones that are often considered to be collocational without over-fitting the data.

5 Preliminary Evaluation

We performed an preliminary evaluation using the first 8939 Japanese sentences in a year's volume of newspaper articles(Mainichi, 1993). We first automatically segmented and tagged these sentences and then revised them by hand. The total number of words in the hand-revised corpus was 226162. We trained our tag models on the corpora with every tenth sentence removed (starting with the first sentence) and then tested the removed sentences. There were 22937 words in the test corpus.

As the first milestone of performance, we tested a hand-crafted tag model of JUMAN (Kurohashi et al., 1994), the most widely used Japanese part-of-speech tagger. The tagging accuracy of JUMAN for the test corpus was only 92.0 %. This shows that our corpus is difficult to tag because the corpus contains various genres of texts; from obituaries to poetry.

Next, we compared the mixture of bi-grams and the mixture of hierarchical tag context trees. In this experiment, only post-positional particles and auxiliaries were *word-level* elements of *basic tags* and all other elements were *subdivision* level. In contrast, bi-gram was constructed by using *subdivision* level. We set the iteration number T to 5. The results of our experiments are summarized in Figure 4.

As a single tree estimator (Number of Mixture = 1), the hierarchical tag context tree attained 94.1 % accuracy, while bi-gram yielded 93.1 %. A hierarchical tag context tree offers a slight improvement, but

Initialize weight[j] = 1 for all examples j
$t = 1$
call *construct-btree*
do
  Read tth example $x_t(< p_t, d_t, w_t >)$
  in which p_t, d_t and w_t represent part-of-speech, subdivision and word, respectively.
  Follow $x_{t-1}, x_{t-2}, \cdots, x_{t-(i-1)}$ and Reach leaf node s
  $low = sw_{t-i}$, $high = sd_{t-i}$
  while($max(\Delta(low), \Delta(high)) \geq Threshold$) {
    if($\Delta(low) \geq \Delta(high)$)
      Expand the tree by the node *low*
    else if(high==sp_{t-i})
      Expand the tree by the node *high*
    else low = sd_{t-i}, high = sp_{t-i}
  }
  $t = t + 1$
while(x_t is not empty)

Table 1: Algorithm *construct-htree*

Input: sequence of N examples $< p_1, d_1, w_1 >, \ldots, < p_N, d_N, w_N >$
in which p_i, d_i and w_i represent part-of-speech, subdivision and word, respectively.
Initialize the weight vector weight[i] =1 for $i = 1, \ldots, N$
Do for $t = 1, 2, \ldots, T$
  Call *construct-htree* providing it with the weight vector weight[] and
  Construct a part-of-speech tagger h_t
  Let *Error* be a set of examples that are not identified by h_t
  Compute the error rate of h_t: $\epsilon_t = \sum_{i \in Error} weight[i] / \sum_{i=1}^{N} weight[i]$
  $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
  For examples correctly predicted by h_t, update the weights vector to be
    weight[i] = weight[i]β_t (4)
Output a final tag model
$$h_f = \sum_{t=1}^{T} (log \frac{1}{\beta_t}) h_t / \sum_{t=1}^{T} (log \frac{1}{\beta_t}) \quad (5)$$

Table 2: Algorithm *mistake-driven mixture*

not a gret deal. This conclusion agrees with Schütze and Singer's experiments that used a context tree of usual part-of-speech.

When we turn to the mixture estimator, a great difference is seen between hierarchical tag context trees and bi-grams. The hierarchical tag context trees produced by the *mistake-driven mixture* method, greatly improved the accuracy and overfitting data was not serious. The best and worst performances were 96.1 % (Number of Mixture = 3) and 94.1 % (Number of Mixture = 1), respectively. On the other hand, the performance of the bi-gram mixture was not satisfactory. The best and worst performances were 93.8 % (Number of Mixture = 2) and 90.8 % (Number of Mixture = 5), respectively.

From the result, we may say exceptional connections are well captured by hierarchical context trees but not by bi-grams. Bi-grams of *subdivision* are too general to selectively detect exceptions.

6 Related Work

Although statistical natural language processing has mainly focused on Maximum Likelihood Estimators, (Pereira et al., 1995) proposed a mixture approach to predict next words by using the Context Tree Weighting (CTW) method (Willems et al., 1995). The CTW method computes probability by mixing subtrees in a single context tree in Bayesian fashion. Although the method is very efficient, it cannot be used to construct hierarchical tag context trees.

Various kinds of *re-sampling* techniques have been studied in statistics (Efron, 1979; Efron and Tibshirani, 1993) and machine learning (Breiman, 1996; Hull et al., 1996; Freund and Schapire, 1996a). In particular, the *mistake-driven mixture* algorithm

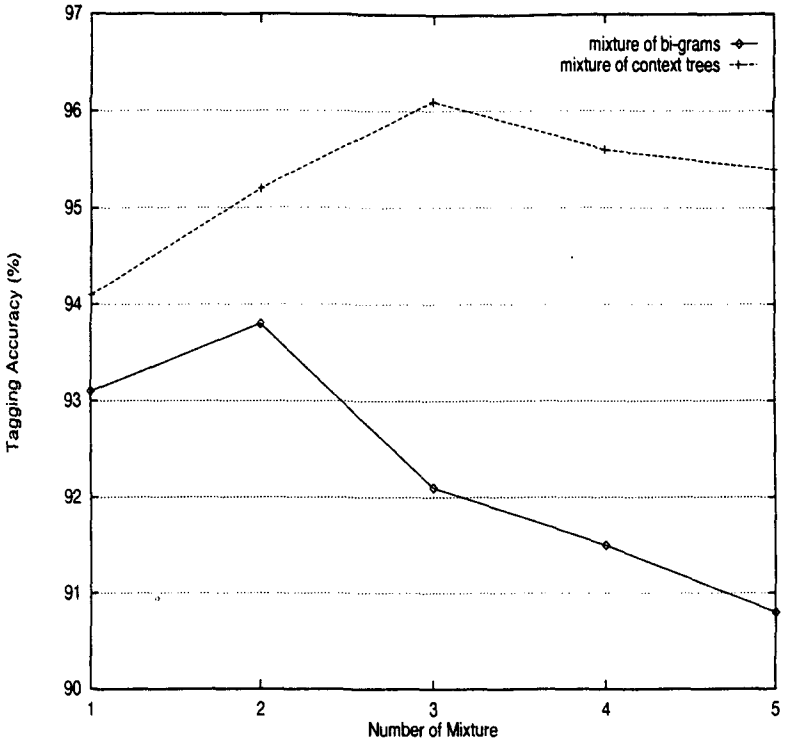

Figure 4: Context Tree Mixture v.s. Bi-gram Mixture

was directly motivated by Adaboost (Freund and Schapire, 1996a). The Adaboost method was designed to construct a high-performance predictor by iteratively calling a weak learning algorithm (that is slightly better than random guess). An empirical work reports that the method greatly improved the performance of decision-tree, k-nearest-neighbor, and other learning methods given relatively simple and sparse data (Freund and Schapire, 1996b). We borrowed the idea of re-sampling to detect exceptional connections and first proved that such a re-sampling method is also effective for a practical application using a large amount of data. The next step is to fill the gap between theory and practition. Most theoretical work on re-sampling assumes *i.i.d* (identically, independently distributed) samples. This is not a realistic assumption in part-of-speech tagging and other NL applications. An interesting future research direction is to construct a theory that handles Markov processes.

7 Conclusion

We have described a new tag model that uses *mistake-driven mixture* to produce *hierarchical tag context trees* that can deal with exceptional connections whose detection is not possible at part-of-speech level. Our experimental results show that combining *hierarchical tag context trees* with the *mistake-driven mixture* method is extremely effective for *1.* incorporating exceptional connections and *2.* avoiding data over-fitting. Although we have focused on part-of-speech tagging in this paper, the *mistake-driven mixture* method should be useful for other applications because detecting and incorporating exceptions is a central problem in corpus-based NLP. We are now costructing a Japanese dependency parser that employes mistake-driven mixture of decision trees.

References

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140, August.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proc. Third Conference on Applied Natural Language Processing*, pages 152–155.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowits. 1993. Equations for Part-of-Speech Tagging. In *Proc. 11th AAAI*, pages 784–789.

K. W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. ACL 2nd Conference on Applied Natural Language Processing*, pages 126–143.

T.M. Cover and J.A. Thomas, 1991. *Elements of Information Theory.* John Wiley & Sons.

B. Efron and R. Tibshirani, 1993. *An Introduction to the Bootstrap.* Chapman and Hall.

B. Efron. 1979. Bootstrap: another look at the jackknife. *The Annals of Statistics*, 7(1):1–26.

Yoav Freund and Robert Schapire. 1996a. A decision-theoretic generalization of on-line learning and an application to boosting.

Yoav Freund and Robert Schapire. 1996b. Experiments with a New Boosting algorithm. In *Proc. 13rd International Conference on Machine Learning*, pages 148–156.

David A. Hull, Jan O. Pedersen, and Hinrich Schütze. 1996. Method combination for document filtering. In *Proc. ACM SIGIR 96*, pages 279–287.

J. Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer juman. In *Proc. International Workshop on Sharable Natural Language Resources*, pages 22–28.

Mainichi, 1993. *CD Mainichi Shinbun.* Nichigai Associates Co.

Masaaki Nagata. 1994. A Stochastic Japanese Morphological Analyzer Using Forward-DP Backward-A^* N-Best Search Algorithm. In *Proc. 15th COLING*, pages 201–207.

Fernando C. Pereira, Yoram Singer, and Naftali Tishby. 1995. Beyond Word N-Grams. In *Proc. Third Workshop on Very Large Corpora*, pages 95–106.

Jorma Rissanen. 1983. A universal data compression system. *IEEE Transaction on Information Theory*, 29(5):656–664, September.

Dana Ron, Yoram Singer, and Naftali Tishby. 1997. The power of amnesia: Learning probabilistic automata with variable memory length. (to appear) Machine Learning Special Issue on COLT94.

H. Schütze and Y. Singer. 1994. Part-of-speech tagging using a variable markov model. In *the 32th Annual Meeting of ACL*, pages 181–187.

M J. Weinberger, J J. Rissanen, and M. Feder. 1995. A universal finite memory source. *IEEE Transaction on Information Theory*, 41(3):643–652, May.

F M J. Willems, Y M. Shtarkov, and T J. Tjalkens. 1995. The context-tree weigting method: Basic properties. *IEEE Transaction on Information Theory*, 41(3):653–664, May.