Tagging English Text with a Probabilistic Model

Bernard Merialdo*†
Institut EURECOM

In this paper we present some experiments on the use of a probabilistic model to tag English text, i.e. to assign to each word the correct tag (part of speech) in the context of the sentence. The main novelty of these experiments is the use of untagged text in the training of the model. We have used a simple triclass Markov model and are looking for the best way to estimate the parameters of this model, depending on the kind and amount of training data provided. Two approaches in particular are compared and combined:

- *using text that has been tagged by hand and computing relative frequency counts,*
- *using text without tags and training the model as a hidden Markov process, according to a Maximum Likelihood principle.*

Experiments show that the best training is obtained by using as much tagged text as possible. They also show that Maximum Likelihood training, the procedure that is routinely used to estimate hidden Markov models parameters from training data, will not necessarily improve the tagging accuracy. In fact, it will generally degrade this accuracy, except when only a limited amount of hand-tagged text is available.

1. Introduction

A lot of effort has been devoted in the past to the problem of tagging text, i.e. assigning to each word the correct tag (part of speech) in the context of the sentence. Two main approaches have generally been considered:

- rule-based (Klein and Simmons 1963; Brodda 1982; Paulussen and Martin 1992; Brill et al. 1990)
- probabilistic (Bahl and Mercer 1976; Debili 1977; Stolz, Tannenbaum, and Carstensen 1965; Marshall 1983; Leech, Garside, and Atwell 1983; Derouault and Merialdo 1986; DeRose 1988; Church 1989; Beale 1988; Marcken 1990; Merialdo 1991; Cutting et al. 1992).

More recently, some work has been proposed using neural networks (Benello, Mackie, and Anderson 1989; Nakamura and Shikano 1989).

* Multimedia Communications Department, Institut EURECOM, 2229 Route des Cretes, B.P. 193, 06904 Valbonne Cedex France; merialdo@eurecom.fr.

† This work was carried out while the author was a visitor of the Continuous Speech Recognition group, IBM T. J. Watson Research Center, Yorktown Heights, NY (USA). Part of the material included in this work has been presented at the *IEEE International Conference on Acoustics, Speech and Signal Processing*, Toronto (Canada), May 1991.

Through these different approaches, some common points have emerged:

- For any given word, only a few tags are possible, a list of which can be found either in the dictionary or through a morphological analysis of the word.
- When a word has several possible tags, the correct tag can generally be chosen from the local context, using contextual rules that define the valid sequences of tags. These rules may be given priorities so that a selection can be made even when several rules apply.

These kinds of considerations fit nicely inside a probabilistic formulation of the problem (Beale 1985; Garside and Leech 1985), which offers the following advantages:

- a sound theoretical framework is provided
- the approximations are clear
- the probabilities provide a straightforward way to disambiguate
- the probabilities can be estimated automatically from data.

In this paper we present a particular probabilistic model, the triclass model, and results from experiments involving different ways to estimate its parameters, with the intention of maximizing the ability of the model to tag text accurately. In particular, we are interested in a way to make the best use of untagged text in the training of the model.

2. The Problem of Tagging

We suppose that the user has defined a set of tags (attached to words). Consider a sentence $W = w_1 w_2 \dots w_n$, and a sequence of tags $T = t_1 t_2 \dots t_n$, of the same length. We call the pair (W, T) an *alignment*. We say that word w_i has been assigned the tag t_i in this alignment.

We assume that the tags have some linguistic meaning for the user, so that among all possible alignments for a sentence there is a **single** one that is correct from a grammatical point of view.

A tagging procedure is a procedure ϕ that selects a sequence of tags (and so defines an alignment) for each sentence.

$$\phi : W \rightarrow T = \phi(W)$$

There are (at least) two measures for the quality of a tagging procedure:

- at sentence level

$$perf_S(\phi) = \text{ percentage of sentences correctly tagged}$$

- at word level

$$perf_W(\phi) = \text{percentage of words correctly tagged}$$

In practice, performance at sentence level is generally lower than performance at word level, since all the words have to be tagged correctly for the sentence to be tagged correctly.

The standard measure used in the literature is performance at word level, and this is the one considered here.

3. Probabilistic Formulation

In the probabilistic formulation of the tagging problem we assume that the alignments are generated by a probabilistic model according to a probability distribution:

$$p(W,T)$$

In this case, depending on the criterion that we choose for evaluation, the optimal tagging procedure is as follows:

- for evaluation at sentence level, choose the most probable sequence of tags for the sentence

  $$\phi(W) = \operatorname*{argmax}_T p(T/W) = \operatorname*{argmax}_T p(W,T)$$

  We call this procedure **Viterbi tagging.** It is achieved using a dynamic programming scheme.

- for evaluation at word level, choose the most probable tag for each word in the sentence

  $$\phi(W)_i = \operatorname*{argmax}_t p(t_i = t/W) = \operatorname*{argmax}_t \sum_{T:t_i=t} p(W,T)$$

  where $\phi(W)_i$ is the tag assigned to word w_i by the tagging procedure ϕ in the context of the sentence W. We call this procedure **Maximum Likelihood (ML) tagging.**

It is interesting to note that the most commonly used method is Viterbi tagging (see DeRose 1988; Church 1989) although it is not the optimal method for evaluation at word level. The reasons for this preference are presumably that:

- Viterbi tagging is simpler to implement than ML tagging and requires less computation (although they both have the same asymptotic complexity)
- Viterbi tagging provides the best interpretation for the sentence, which is linguistically appealing
- ML tagging may produce sequences of tags that are linguistically impossible (because the choice of a tag depends on all contexts taken together).

However, in our experiments, we will show that Viterbi and ML tagging result in very similar performance.

Of course, the real tags have not been generated by a probabilistic model and, even if they had been, we would not be able to determine this model exactly because of practical limitations. Therefore the models that we construct will only be approximations of an ideal model that does not exist. It so happens that despite these assumptions and approximations, these models are still able to perform reasonably well.

4. The Triclass Model

We have the mathematical expression:

$$p(W,T) = \prod_{i=1}^{n} p(w_i/w_1t_1 \ldots w_{i-1}t_{i-1}t_i).p(t_i/w_1t_1 \ldots w_{i-1}t_{i-1})$$

The triclass (or tri-POS [Derouault 1986], or tri-Ggram [Codogno et al. 1987], or HK) model is based on the following approximations:

- The probability of the tag given the past depends only on the last two tags
$$p(t_i/w_1t_1 \ldots w_{i-1}t_{i-1}) = h(t_i/t_{i-2}t_{i-1})$$
- The probability of the word given the past depends only on its tag
$$p(w_i/w_1t_1 \ldots w_{i-1}t_{i-1}t_i) = k(w_i/t_i)$$

(the name HK model comes from the notation chosen for these probabilities).

In order to define the model completely we have to specify the values of all h and k probabilities. If N_W is the size of the vocabulary and N_T the number of different tags, then there are:

- $N_T.N_T.N_T$ values for the h probabilities
- $N_W.N_T$ values for the k probabilities.

Also, since all probability distributions have to sum to one, there are:

- $N_T.N_T$ equations to constrain the values for the h probabilities
- N_T equations to constrain the values for the k probabilities.

The total number of free parameters is then:

$$(N_W - 1).N_T + (N_T - 1).N_T.N_T.$$

Note that this number grows only linearly with respect to the size of the vocabulary, which makes this model attractive for vocabularies of a very large size.

The triclass model by itself allows any word to have any tag. However, if we have a dictionary that specifies the list of possible tags for each word, we can use this information to constrain the model: if t is not a valid tag for the word w, then we are sure that

$$k(w/t) = 0.$$

There are thus at most as many nonzero values for the k probabilities as there are possible pairs (word, tag) allowed in the dictionary.

5. Training the Triclass Model

We consider two different types of training:

- Relative Frequency (RF) training
- Maximum Likelihood (ML) training which is done via the Forward-Backward (FB) algorithm.

5.1 Relative Frequency Training

If we have some tagged text available we can compute the number of times $N(w,t)$ a given word w appears with the tag t, and the number of times $N(t_1,t_2,t_3)$ the sequence (t_1,t_2,t_3) appears in this text. We can then estimate the probabilities h and k by computing the relative frequencies of the corresponding events on this data:

$$h_{rf}(t_3/t_1,t_2) = f(t_3/t_1,t_2) = \frac{N(t_1,t_2,t_3)}{N(t_1,t_2)}$$

$$k_{rf}(w/t) = f(w/t) = \frac{N(w,t)}{N(t)}$$

These estimates assign a probability of zero to any sequence of tags that did not occur in the training data. But such sequences may occur if we consider other texts. A probability of zero for a sequence creates problems because any alignment that contains this sequence will get a probability of zero. Therefore, it may happen that, for some sequences of words, all alignments get a probability of zero and the model becomes useless for such sentences.

To avoid this, we interpolate these distributions with uniform distributions, i.e. consider the interpolated model defined by:

$$h_{inter}(t_3/t_1,t_2) = \lambda.h_{rf}(t_3/t_1,t_2) + (1-\lambda).h_{unif}(t_3/t_1,t_2)$$

$$k_{inter}(w/t) = \lambda.k_{rf}(w/t) + (1-\lambda).k_{unif}(w/t)$$

where

$$h_{unif}(t_3/t_1,t_2) = \frac{1}{N_T}$$

$$k_{unif}(w/t) = \frac{1}{\text{number of words that have the tag t}}$$

The interpolation coefficient λ is computed using the deleted interpolation algorithm (Jelinek and Mercer 1980) (it would also be possible to use two coefficients, one for the interpolation on h, one for the interpolation on k). The value of this coefficient is expected to increase if we increase the size of the training text, since the relative frequencies should be more reliable. This interpolation procedure is also called "smoothing."

Smoothing is performed as follows:

- Some quantity of tagged text from the training data is not used in the computation of the relative frequencies. It is called the "held-out" data.
- The coefficient λ is chosen to maximize the probability of emission of the held-out data by the interpolated model.

- This maximization can be performed by the standard Forward-Backward (FB) or Baum–Welch algorithm (Baum and Eagon 1967; Jelinek 1976; Bahl, Jelinek, and Mercer 1983; Poritz 1988), by considering λ and $1 - \lambda$ as the transition probabilities of a Markov model.

It can be noted that more complicated interpolation schemes are possible. For example, different coefficients can be used depending on the count of (t_1, t_2), with the intuition that relative frequencies can be trusted more when this count is high. Another possibilitity is to interpolate also with models of different orders, such as $h_{rf}(t_3/t_2)$ or $h_{rf}(t_3)$.

Smoothing can also be achieved with procedures other than interpolation. One example is the "backing-off" strategy proposed by Katz (1987).

5.2 Maximum Likelihood Training

Using a triclass model M it is possible to compute the probability of any sequence of words W according to this model:

$$p_M(W) = \sum_T p_M(W, T)$$

where the sum is taken over all possible alignments. The Maximum Likelihood (ML) training finds the model M that maximizes the probability of the training text:

$$\max_M \prod_W p_M(W)$$

where the product is taken over all the sentences W in the training text. This is the problem of training a hidden Markov model (it is hidden because the sequence of tags is hidden). A well-known solution to this problem is the Forward-Backward (FB) or Baum–Welch algorithm (Baum and Eagon 1967; Jelinek 1976; Bahl, Jelinek, and Mercer 1983), which iteratively constructs a sequence of models that improve the probability of the training data.

The advantage of this approach is that it does not require any tagging of the text, but makes the assumption that the correct model is the one in which tags are used to best predict the word sequence.

6. Tagging Algorithms

The Viterbi algorithm is easily implemented using a dynamic programming scheme (Bellman 1957). The Maximum Likelihood algorithm appears more complex at first glance, because it involves computing the sum of the probabilities of a large number of alignments. However, in the case of a hidden Markov model, these computations can be arranged in a way similar to the one used during the FB algorithm, so that the overall amount of computation needed becomes linear in the length of the sentence (Baum and Eagon 1967).

7. Experiments

The main objective of this paper is to compare RF and ML training. This is done in Section 7.2. We also take advantage of the environment that we have set up to perform other experiments, described in Section 7.3, that have some theoretical interest, but did

Table 1
RF training on N sentences, Viterbi tagging.

Training data (sentences)	Interpolation coefficient λ	Nb of errors (words)	% correct tags
0	.0	10498	77.0
100	.48	4568	90.0
2000	.77	2110	95.4
5000	.85	1744	96.2
10000	.90	1555	96.6
20000	.92	1419	96.9
all	.94	1365	97.0

not bring any improvement in practice. One concerns the difference between Viterbi and ML tagging, and the other concerns the use of constraints during training.

We shall begin by describing the textual data that we are using, before presenting the different tagging experiments using these various training and tagging methods.

7.1 Text Data

We use the "treebank" data described in Beale (1988). It contains 42,186 sentences (about one million words) from the Associated Press. These sentences have been tagged manually at the Unit for Computer Research on the English Language (University of Lancaster, U.K.), in collaboration with IBM U.K. (Winchester) and the IBM Speech Recognition group in Yorktown Heights (USA). In fact, these sentences are not only tagged but also parsed. However, we do not use the information contained in the parse.

In the treebank 159 different tags are used. These tags were projected on a smaller system of 76 tags designed by Evelyne Tzoukermann and Peter Brown (see Appendix). The results quoted in this paper all refer to this smaller system.

We built a dictionary that indicates the list of possible tags for each word, by taking all the words that occur in this text and, for each word, all the tags that are assigned to it somewhere in the text. In some sense, this is an optimal dictionary for this data, since a word will not have all its possible tags (in the language), but only the tags that it actually had within the text.

We separated this data into two parts:

- a set of 40,186 tagged sentences, the **training data**, which is used to build the models
- a set of 2,000 tagged sentences (45,583 words), the **test data**, which is used to test the quality of the models.

7.2 Basic Experiments

RF training, Viterbi tagging

In this experiment, we extracted N tagged sentences from the training data. We then computed the relative frequencies on these sentences and built a "smoothed" model using the procedure previously described. This model was then used to tag the 2,000 test sentences. We experimented with different values of N, for each of which we indicate the value of the interpolation coefficient and the number and percentage of correctly tagged words. Results are indicated in Table 1.

As expected, as the size of the training increases, the interpolation coefficient increases and the quality of the tagging improves.

When $N = 0$, the model is made up of uniform distributions. In this case, all alignments for a sentence are equally probable, so that the choice of the correct tag is just a choice at random. However, the percentage of correct tags is relatively high (more than three out of four) because:

- almost half of the words of the text have a single possible tag, so that no mistake can be made on these words
- about a quarter of the words of the text have only two possible tags so that, on the average, a random choice is correct every other time.

Note that this behavior is obviously very dependent on the system of tags that is used.

It can be noted that reasonable results are obtained quite rapidly. Using 2,000 tagged sentences (less than 50,000 words), the tagging error rate is already less than 5%. Using 10 times as much data (20,000 tagged sentences) provides an improvement of only 1.5%.

ML training, Viterbi tagging

In ML training we take all the training data available (40,186 sentences) but we only use the word sequences, not the associated tags (except to compute the initial model, as will be described later). This is possible since the FB algorithm is able to train the model using the word sequence only.

In the first experiment we took the model made up of uniform distributions as the initial one. The only constraints in this model came from the values $k(w/t)$ that were set to zero when the tag t was not possible for the word w (as found in the dictionary). We then ran the FB algorithm and evaluated the quality of the tagging. The results are shown in Figure 1. (Perplexity is a measure of the average branching factor for probabilistic models.)

This figure shows that ML training both improves the perplexity of the model and reduces the tagging error rate. However, this error rate remains at a relatively high level—higher than that obtained with a RF training on 100 tagged sentences.

Having shown that ML training is able to improve the uniform model, we then wanted to know if it was also able to improve more accurate models. We therefore took as the initial model each of the models obtained previously by RF training and, for each one, performed ML training using all of the training word sequences. The results are shown graphically in Figure 2 and numerically in Table 2.

These results show that, when we use few tagged data, the model obtained by relative frequency is not very good and Maximum Likelihood training is able to improve it. However, as the amount of tagged data increases, the models obtained by Relative Frequency are more accurate and Maximum Likelihood training improves on the initial iterations only, but after deteriorates. If we use more than 5,000 tagged sentences, even the first iteration of ML training degrades the tagging. (This number is of course dependent on both the particular system of tags and the kind of text used in this experiment).

These results call for some comments. ML training is a theoretically sound procedure, and one that is routinely and successfully used in speech recognition to estimate the parameters of hidden Markov models that describe the relations between sequences of phonemes and the speech signal. Although ML training is guaranteed to improve perplexity, perplexity is not necessarily related to tagging accuracy, and it is possible to improve one while degrading the other. Also, in the case of tagging,

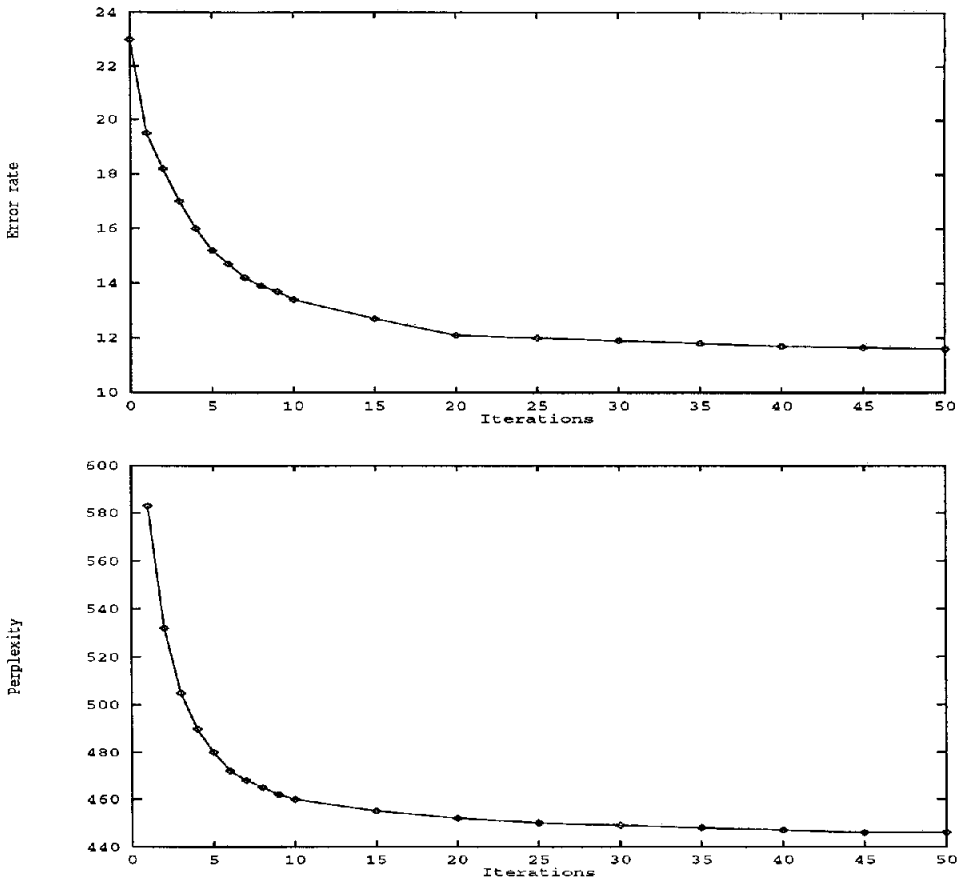

Figure 1
ML training from uniform distributions.

Table 2
ML training from various initial points.

	Number of tagged sentences used for the initial model						
	0	100	2000	5000	10000	20000	all
Iter	Correct tags (% words) after ML on 1M words						
0	77.0	90.0	95.4	96.2	96.6	96.9	97.0
1	80.5	92.6	95.8	96.3	96.6	96.7	96.8
2	81.8	93.0	95.7	96.1	96.3	96.4	96.4
3	83.0	93.1	95.4	95.8	96.1	96.2	96.2
4	84.0	93.0	95.2	95.5	95.8	96.0	96.0
5	84.8	92.9	95.1	95.4	95.6	95.8	95.8
6	85.3	92.8	94.9	95.2	95.5	95.6	95.7
7	85.8	92.8	94.7	95.1	95.3	95.5	95.5
8	86.1	92.7	94.6	95.0	95.2	95.4	95.4
9	86.3	92.6	94.5	94.9	95.1	95.3	95.3
10	86.6	92.6	94.4	94.8	95.0	95.2	95.2

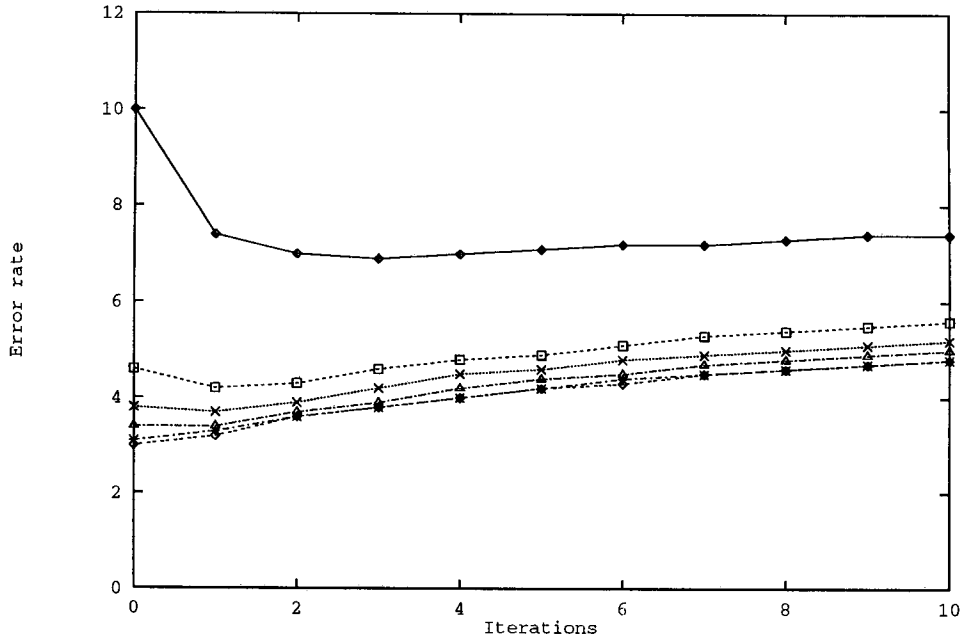

Figure 2
ML training from various initial points *(top line corresponds to N=100, bottom line to N=all).*

the relations between words and tags are much more precise than the relations between phonemes and speech signals (where the correct correspondence is harder to define precisely). Some characteristics of ML training, such as the effect of smoothing probabilities, are probably more suited to speech than to tagging.

7.3 Extra Experiments

Viterbi versus ML tagging

For this experiment we considered the initial model built by RF training over the whole training data and all the successive models created by the iterations of ML training. For each of these models we performed Viterbi tagging and ML tagging on the same test data, then evaluated and compared the number of tagging errors produced by these two methods. The results are shown in Table 3.

The models obtained at different iterations are related, so one should not draw strong conclusions about the definite superiority of one tagging procedure. However, the difference in error rate is very small, and shows that the choice of the tagging procedure is not as critical as the kind of training material.

Constrained ML training

Following a suggestion made by F. Jelinek, we investigated the effect of constraining the ML training by imposing constraints on the probabilities. This idea comes from the observation that the amount of training data needed to properly estimate the model increases with the number of free parameters of the model. In the case of little training data, adding reasonable constraints on the shape of the models that are looked for reduces the number of free parameters and should improve the quality of the estimates.

Table 3
Viterbi vs. ML tagging.

Iter.	Tagging errors out of 45,583 words				
	Viterbi		ML		Vit. - ML
0	%	nb	%	nb	nb
0	97.01	1365	97.01	1362	3
1	96.76	1477	96.75	1480	−3
2	96.44	1623	96.47	1607	16
3	96.23	1718	96.23	1719	−1
4	96.00	1824	96.02	1812	12
5	95.82	1906	95.85	1892	14
6	95.66	1978	95.68	1970	8
7	95.51	2046	95.54	2031	15
8	95.39	2100	95.42	2087	13
9	95.30	2144	95.31	2140	4
10	95.21	2183	95.22	2177	6

Table 4
Standard ML vs. tw-constrained ML training.

Iter.	Tagging errors out of 45,583 words			
	ML		tw-c. ML	
0	%	nb	%	nb
0	97.01	1365	97.01	1365
1	96.76	1477	96.87	1427
2	96.44	1623	96.71	1501
3	96.23	1718	96.57	1562
4	96.00	1824	96.43	1626
5	95.82	1906	96.36	1661
6	95.66	1978	96.29	1690
7	95.51	2046	96.22	1723
8	95.39	2100	96.18	1741
9	95.30	2144	96.12	1768
10	95.21	2183	96.09	1784

We tried two different constraints:

- The first one keeps $p(t/w)$ fixed if w is a frequent word, in our case one of the 1,000 most frequent words. We call it *tw-constraint*. The rationale is that if w is frequent, the relative frequency provides a good estimate for $p(t/w)$ and the training should not change it.
- The second one keeps the marginal distribution $p(t)$ constant and is based on a similar reasoning. We call it *t-constraint*.

tw-constraint
The tw-constrained ML training is similar to the standard ML training, except that the probabilities $p(t/w)$ are not changed at the end of an iteration.

The results in Table 4 show the number of tagging errors when the model is trained with the standard or tw-constrained ML training. They show that the tw-constrained ML training still degrades the RF training, but not as quickly as the standard ML. We

Table 5
Standard ML vs. constrained ML training.

Iter.	Tagging errors out of 45,583 words (biclass model)			
	ML		t-c. ML	
0	%	nb	%	nb
0	96.87	1429	96.87	1429
1	96.51	1592	96.54	1576
2	96.18	1743	96.23	1718
3	96.00	1824	96.03	1810
4	95.84	1896	95.90	1871
5	95.67	1972	95.77	1928
6	95.52	2044	95.59	2009
7	95.42	2087	95.50	2051
8	95.33	2129	95.42	2087
9	95.24	2171	95.34	2126
10	95.18	2196	95.30	2141

have not tested what happens when smaller training data is used to build the initial model.

t-constraint
This constraint is more difficult to implement than the previous one because the probabilities $p(t)$ are not the parameters of the model, but a combination of these parameters. With the help of R. Polyak we have designed an iterative procedure that allows the likelihood to be improved while preserving the values of $p(t)$. We do not have sufficient space to describe this procedure here. Because of its greater computational complexity, we have only applied it to a biclass model, i.e. a model where

$$p(t_i/w_1t_1 \ldots w_{i-1}t_{i-1}) = h(t_i/t_{i-1}).$$

The initial model is estimated by relative frequency on the whole training data and Viterbi tagging is used.

As in the previous experiment, the results in Table 5 show the number of tagging errors when the model is trained with the standard or t-constrained ML training. They show that the t-constrained ML training still degrades the RF training, but not as quickly as the standard ML. Again, we have not tested what happens when smaller training data is used to build the initial model.

8. Conclusion

The results presented in this paper show that estimating the parameters of the model by counting relative frequencies over a very large amount of hand-tagged text lead to the best tagging accuracy.

Maximum Likelihood training is guaranteed to improve perplexity, but will not necessarily improve tagging accuracy. In our experiments, ML training degrades the performance unless the initial model is already very bad.

The preceding results suggest that the optimal strategy to build the best possible model for tagging is the following:

- get as much tagged (by hand) text as you can afford

- compute the relative frequencies from this data to build an initial model M_0
- get as much untagged text as you can afford
- starting from M_0, perform the Forward-Backward iterations. At each iteration, evaluate the tagging quality of the new model M_i on some held-out tagged text. Stop either when you have reached a preset number of iterations or the model M_i performs worse than M_{i-1}, whichever occurs first.

Acknowledgments

I would like to thank Peter Brown, Fred Jelinek, John Lafferty, Robert Mercer, Salim Roukos, and other members of the Continuous Speech Recognition group for the fruitful discussions I had with them throughout this work. I also want to thank one of the referees for his judicious comments.

References

Bahl, Lalit R., and Mercer, Robert L. (1976). "Part of speech assignment by a statistical decision algorithm." In *IEEE International Symposium on Information Theory*, 88–89. Ronneby.

Bahl, Lalit R.; Jelinek, Frederick and Mercer, Robert L. (1983). "A maximum likelihood approach to continuous speech recognition," In *IEEE Transactions on PAMI*, 5(2), 179–190.

Baum, L. E., and Eagon, J. A. (1967). "An inequality with application to statistical estimation for probabilistic functions of Markov processes and to a model for ecology." *Bulletin of the American Mathematicians Society*, 73, 360–363.

Beale, A. D. (1985). "A probabilistic approach to grammatical analysis of written English by computer." In *Proceedings, Second Conference of the European Chapter of the ACL*, Geneva, Switzerland, 159–165.

Beale, A. D. (1988). "Lexicon and grammar in probabilistic tagging of written English." In *Proceedings, 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo NY: 211–216.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Benello, J.; Mackie, A. W.; and Anderson, J. A. (1989). "Syntactic category disambiguation with neural networks." *Computer Speech and Language*, 3, 203–217.

Brill, E.; Magerman, D.; Marcus, M.; and Santorini, B. (1990). "Deducing linguistic structure from the statistics of large corpora." In *Proceedings, DARPA Speech and Natural Language Workshop*, Hidden Valley PA. 275–282.

Brodda, Benny (1982). "Problems with tagging and a solution." *Nordic Journal of Linguistics*, 93–116.

Church, Kenneth W. (1989). "A stochastic parts program noun phrase parser for unrestricted text." In *IEEE Proceedings of the ICASSP*, Glasgow, 695–698.

Codogno, M.; Fissore, L.; Martelli, A.; Pirani, G.; and Volpi, G. (1987). "Experimental evaluation of Italian language models for large-dictionary speech recognition." In *Proceedings, European Conference on Speech Technology*, Edinburgh, 159–162.

Cutting, D.; Kupiec, J.; Pedersen, J.; and Sibun, P. (1992). "A practical part-of-speech tagger." In *Proceedings, Third Conference on Applied Language Processing*, Trento, Italy, 133–140.

Debili, Fathi (1977). "Traitements syntaxiques utilisant des matrices de precedence frequentielles construites automatiquement par apprentissage." Doctoral dissertation, Engineering Department, Universite Paris 7, France.

DeRose, S. (1988). "Grammatical category disambiguation by statistical optimization." *Computational Linguistics*, 14(1), 31–39.

Derouault, Anne-Marie, and Merialdo, Bernard (1986). "Natural language modeling for phoneme-to-text transcription." In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 742–749.

Garside, R., and Leech, F. (1985). "A probabilistic parser." In *Proceedings, Second Conference of the European Chapter of the ACL*, Geneva, Switzerland, 166–170.

Jelinek, Frederick (1976). "Continuous speech recognition by statistical methods." In *Proceedings of the IEEE*, 64, 532–556.

Jelinek, Frederick, and Mercer, Robert L.

(1980). "Interpolated estimation of Markov source parameters from sparse data." In *Proceedings, Workshop on Pattern Recognition in Practice*, Amsterdam, 381–397.

Katz, S. (1987). "Estimation of probabilities from sparse data for the language model component of a speech recognizer." *IEEE Transactions on ASSP*, 34(3), 400–401.

Klein, S., and Simmons, R. F. (1963). "A grammatical approach to grammatical coding of English words." *JACM*, 10, 334–347.

Leech, G.; Garside, R.; and Atwell, E. (1983). "The automatic grammatical tagging of the LOB corpus." *Newsletter of the International Computer Archive of Modern English*, 7, 13–33.

de Marcken, C. G. (1990). "Parsing the LOB corpus." In *Proceedings, ACL Annual Meeting*, Pittsburg PA, 243–251.

Marshall, Ian (1983). "Choice of grammatical word-class without global syntactic analysis: Tagging words in the LOB corpus." *Computers and the Humanities*, 139–150.

Merialdo, Bernard (1991). "Tagging text with a probabilistic model." In *IEEE Proceedings of the ICASSP*, Toronto, 809–812.

Nakamura, M., and Shikano, K. (1989). "A study of English word category prediction based on neural networks." In *IEEE Proceedings of the ICASSP*, Glasgow, 731–734.

Paulussen, H., and Martin, W. (1992). "Dilemma-2: A lemmatizer-tagger for medical abstracts." In *Proceedings, Third Conference on Applied Language Processing*, Trento, Italy, 141–146.

Poritz, Alan B. (1988). "Hidden Markov models: A guided tour." In *IEEE Proceedings of the ICASSP*, New York, 7–13.

Stolz, W. S.; Tannenbaum, P. H.; and Carstensen F. V. (1965). "A stochastic approach to the grammatical coding of English." *Communications of the ACM*, 8, 399–405.

Appendix A: List of Tags Used

$* possessive marker (*'s, '*)

APP$* possessive adjectives (*my, your, our*)

AT* article (*the, a, no*)

BOUNDARY_TAG end-of-sentence marker

CCF* coordinating conjunction (*and, or, but, so, yet, then*)

CS* subordinating conjunction (*if, because, unless*)

CT* *that* or *whether* as subordinating conjunctions

D* determiner (*all, any, enough*)

D*Q wh-determiner (*which, what, whose*)

D*R comparative plural after-determiner (*less, more*)

D*1 determiner singular (*this, that, little, much, former*)

D*2 determiner plural (*these, few, several, many*)

DAT* superlative determiner (*least, most*)

EX* existential *there*

FW* foreign words (*ipso, facto*)

I* preposition (*general*)

ICS* preposition that can also be used as a conjunction (*since, after*)

IF* the preposition *for*

IO* the preposition *of*

J* adjective (*small, pretty*)

J*R comparative adjective (*smaller, prettier*)

J*T superlative adjective (*prettiest, nicest*)

LE* leading coordinator (*both, either, neither*)

M* cardinal number

MD* ordinal number (*first, second*)

N* noun without number (*english*)

N*1 singular noun (*cat, man*)

N*2 plural noun (*cats, men*)

NPR* proper noun (*paris, fred*)

NR* noun/adverb of direction (*south, west*) or time (*now, tomorrow, tuesday*)

P* non-nominative pronoun (*none, anyone, oneself*)

P*Q *who, whom, whoever, whomever*

PNX1* personal pronoun reflexive (*himself*)

PN1* indefinite pronoun (*anyone, anybody*)

PP$* possessive pronoun (*mine, yours*)

PP*O personal pronoun object (*me, him*)

PP*S personal pronoun subject (*I, you, we*)

PP*S3 personal pronoun subject 3rd person singular (*he, she*)

PUNCT1* end of sentence (. ! ? –)

PUNCT2* non terminal punctuation (, : ;)

QUOT* quote

R* adverb (*here, slowly*)

R*Q wh-adverb (*where, when, why, how, whenever, wherever*)

R*R comparative adverb (*better, longer*)

RG* degree adverb (*very, so, too, enough, indeed*)

RGQ* wh-degree adverb (*how*)

RGR* comparative degree adverb (*more, less, worse*)

RP* adverb that can also serve as a preposition

SIGN* sign ($, *c., ct,* %)

TO* *to* as pre-infinitive

UH* interjection (*gee*)

VBDR* *were*

VBDZ* *was*

VBG* *being*

VBI* infinitive form of *be* and imperative

VBM* *am*

VBN* *been*

VBR* *are*

VBZ* *is*

VDG* *doing*

VDN* past participial form of *do* (*did*)

VDPAST* past form of *do* (*did*)

VD0* *do* as a conjugated form and infinitive

VD0Z* *does* as a conjugated form

VHG* *having*

VHN* past participial form of have (*had*)

VHPAST* past form of have (*had*)

VH0* *have* as a conjugated form

VH0Z* *has* as a conjugated form

VM* modals (*can, would, ought, used*)

VVG* non-aux verb in *-ing*

VVN* past participial form of non-aux verb

VVPAST* preterit of non-aux verb

VV0* non-third-person-singular form of non-aux verb and infinitive

VV0Z* third-person-singular form of non-aux verb

XX* *not*