Unsupervised Part-of-speech Tagging

Mihai Pop

March 27, 1996

Abstract

Different approaches have been taken in order to solve the part-of-speech tagging problem. Several methods for unsupervised tagging have obtained good accuracies in practice. The approach taken by Brill [Bri95] obtains results comparable to the best existing taggers. In this paper we explore the details of this unsupervised part-of-speech tagger and we present a comparison to the Xerox tagger, which is reportedly the best tagger available at the moment.

1 Introduction

A specific word may belong to different grammatical categories, depending on the context in which it is used. This ambiguity does not seem to be a big problem for humans. We can very easily tell the difference between different meanings of the same word. Knowing the part of speech of a word we are able to discern among several possible meanings of a phrase. It makes sense to assume that providing a computer with part of speech information for all words in a text, would make the computers task of understanding the text much easier. Indeed most parsers (programs that determine the syntactic structure of a text) assume that each word is tagged with its part of speech. We have thus separated the problem of automatically "understanding" text into two problems:

part-of-speech tagging – assign each word in a text a tag indicating its part of speech

parsing – determine the syntactic structure of a tagged text

We will now concentrate on several approaches to part-of-speech tagging.

The main idea in part-of-speech tagging is that a very small context almost always determines the part of speech of a word. Words can usually only belong to a small set of grammatical categories. By using statistical information based on a small context (usually one word before or after the current word) we can determine the correct tag for each word in a text. This approach is called *statistical disambiguation* because it uses statistical information to disambiguate among the set of all possible tags of a word.

2 Approaches to Part-of-Speech Tagging

2.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a stochastic automaton. In the case of part-of-speech tagging, the output symbols are possible tags. The HMM consists of a set of states and a matrix of state transition probabilities. For a sequence of input symbols, the automaton can produce different sequences of output symbols, with different probabilities, depending on the structure of the model and the transition probabilities. We are usually interested in the most likely output, i.e. the most probable sequence of tags.

In order to tag a text, we feed the HMM a sequence of all possible tags for the words in the text, and we obtain the most likely sequence of tags. The computation of the most likely output sequence is done using the Viterbi algorithm (see good description in [Lee89]).

Training an HMM means determining the transition probabilities. The Baum-Welch algorithm (see good description in [Lee89]) adjusts the probabilities so that the probability of the model generating the training sequence is maximized. The initial transition probabilities can be determined using statistical information derived from the training text.

2.2 Transformation Based

Another approach to part-of-speech tagging is the use of transformation-based disambiguation. Recall from section 1 that tagging involves disambiguating sets of tags. This operation can be expressed by rules like: **change tag set $\mathcal{X}$ into tag Y when $\mathcal{P}$ is true**, where $\mathcal{P}$ is a logic predicate.

In most cases, information derived from a small context around the current set of tags is sufficient to determine the correct disambiguation. We can therefore use rules like: **change tag set $\mathcal{X}$ into tag Y when previous word is tagged with tag Z** or variants thereof. In the current work we have used rules referring to the previous or next tag, and to the previous or next word.

Once a set of rules has been determined, applying them is a straight-forward task. The text is being scanned, and for each context matching the context of the current rule, the tag set is modified as specified in the rule. An example of this would be a rule that says **when previous word is a determiner, change a tag set $\mathcal{X} = \{noun, verb\}$ into** *noun*. If the text looks like: `the/det cat/noun_verb`[1] the tagger outputs the tag *noun*. We fire such rules only when both the current context matches the context specified in the rule, and when the current tag set includes the tag set specified by the rule. Therefore, if a certain rule disambiguates **noun_verb** into **verb**, we apply the rule when the current tags are **adverb_noun_verb** but do not apply it when the tags are **adverb_verb**.

We must however be careful since applying a rule to the current tag set may modify the context for the next tag set. This problem does not appear if modifications are made on a temporary copy of the text, which replaces the original text after the application of all the rules. This is the approach taken by our tagger. The other alternative, in which rules are applied directly on the original text, can also be used. In this case we allow for dynamic changes in the text while rules are applied. Maintaining consistent information about word/tag and

[1] the underscore means *or*

word/tag-pair frequencies might make the code more complicated, especially as all changes must be reversed before next rule is evaluated.

Another parameter that has to be considered is the way in which rules are applied, with respect to the tag sets in the text. We could apply all valid rules to the first tag set and only then proceed to the next one, or apply each rule in turn to the whole text. Our tagger uses the later approach.

Training represents determining a set of rules. It is done in a very simple manner. Given a tagged text, we remove all tags, and re-tag the text by assigning each word the set of all possible tags as specified by the lexicon. We will call this initial tagging step "the starting tagging" or "the initial tagging". Using the original tagged text and the initially tagged text, we repeat the following procedure.

- Select the most frequent ambiguous tag set in the initially tagged text;
- Consider all the possible rules matching the required template (e.g. `If word to the left is ... then ...`) and apply them;
- For each rule, measure the distance between the original text and the current text. The rule that brings us closest to the original text is applied and added to the output set;
- Repeat until there are no more ambiguous tags.

The set of rules thus determined can be used to tag previously unseen text by applying the procedure mentioned above.

3 Unsupervised Learning

Once we train our tagger on text from a specific subject area we usually only obtain good results if we run the tagger on texts from the same subject area. In order to switch areas we need manually tagged text from the new domain. It is hard and expensive to obtain such texts. It is therefore useful to determine a method for training a tagger without the need of a manually tagged training corpus. This is called *unsupervised training* because it eliminates user help in tagging the training data. Although it may seem that the information in a "correctly" tagged text is necessary for inferring tagging rules, high tagging accuracies have been obtained when training without previously tagged data. The only information available to an unsupervised tagger is a lexicon associating possible tags to words. This information can be used to initially tag the text by assigning each word the set of all possible tags. Starting from such a text, taggers learn a way to disambiguate the sets of tags in order to obtain an unambiguously tagged text. In the following sections we present different approaches to unsupervised learning.

3.1 Hidden Markov Models

The Baum-Welch algorithm for training HMMs works well even if the training text is the initial tagged text (where each word is tagged with all possible tags). Most of the words in a lexicon are unambiguously tagged. In our lexicon, 33000 words out of 36000 had only one tag. This information can be used by the tagger to infer disambiguation criteria. Therefore, the approach to unsupervised tagging taken by HMMs is:

- Initially tag training text
- Train tagger on initially tagged text
- use HMM obtained during previous step to tag the test text

Using this approach Cutting et al. [CKPS92] report 96% accuracy on the Brown corpus [FK82] [2].

3.2 Transformation Based

Transformation based unsupervised tagging follows a similar approach. The text is initially tagged and then the learner tries to learn transformation rules. In the supervised case it is easy to compare the accuracy gain obtained after applying different rules, by referring to the "correctly" tagged text. When training in an unsupervised manner we need a different criterion to compare possible rules. We use scoring functions which, based on statistical information obtained from the current text, assign each rule a specific score. The rule with the highest score is chosen by the learner. The unsupervised learner then applies the rule and keeps looking for rules until all possible candidates obtain a null score.

Tagging a text after a set of rule was determined is done in the same way as in the supervised case.

The following section presents the scoring functions in more detail.

4 Scoring Functions

The scoring function used by the unsupervised tagger tells the tagger which rule is "better", in the sense of getting closer to a "correctly" tagged text. A rule is given a score of 0 when its application cannot disambiguate any tag sets. Because the learner stops when a rule scores 0, convergence is insured. Due to the finite number of possible tags for each word, after a finite number of steps, either all tag sets have been disambiguated, or no more ambiguous tag sets can be resolved. At this point rules score 0, therefore insuring the termination of the learner.

Although it is hard to decide whether a certain scoring function accurately captures the word–tag relationship, several measures behave well in practice. We describe these measures in the following sections.

[2] In their approach they use an idea originated by Kupiec [Kup92] in which words are grouped in equivalence classes, thus reducing the number of parameters that need to be adjusted

4.1 Paper

The measure presented in [Bri95] uses the frequency information for unambiguous tags to determine the score of applying a specific rule.

Given a rule $(\mathcal{X} \longrightarrow Y)_C$, where $\mathcal{X}$ is a set of tags and C a context, the measure M of the rule is given by the following equations:

$$\begin{aligned} R &= \operatorname*{argmax}_{Z \in \mathcal{X}, Z \neq Y} \left(\frac{\text{freq}(Y)}{\text{freq}(Z)} \cdot \text{incontext}(Z, C) \right) \\ M &= \text{incontext}(Y, C) - \frac{\text{freq}(Y)}{\text{freq}(R)} \cdot \text{incontext}(R, C) \end{aligned}$$

The score is given by the difference between the frequency of the destination tag Y in the given context, and the frequency of the most likely tag, in the same context. The values are adjusted by the overall frequencies of the two tags in order to avoid a strong influence of infrequent tags. All frequencies are determined based on unambiguously tagged words.

4.2 Original

A variant of the previous measure compares all pairs of tags in tag set $\mathcal{X}$. The pair with the highest score determines the score of the rule resolving $\mathcal{X}$ into tag Y.

$$M = \max_{\substack{X, Y \in \mathcal{X} \\ \text{freq}(X) < \text{freq}(Y)}} \left(\frac{\text{freq}(X)}{\text{freq}(Y)} \cdot \text{incontext}(Y, C) - \text{incontext}(X, C) \right)$$

4.3 Logarithmic

Following the same idea of comparing all pairs of tags in a particular tag set, we can use a logarithmic scoring function, similar to the approach of Yarowski [Yar95]. This measure compares the frequencies of the two tags in the current context. As in the previous two measures, the comparison is adjusted by the overall frequencies of the tags.

$$M = \max_{\substack{X, Y \in \mathcal{X} \\ \text{freq}(X) < \text{freq}(Y)}} \log \left(\frac{\text{incontext}(Y, C)}{\text{freq}(Y)} \cdot \frac{\text{freq}(X)}{\text{incontext}(X, C)} \right)$$

4.4 Performance of different scoring functions

We ran tests with different measures and different values of lexicon purity (see section 4.5). All the tests were performed on a test and a training text of $20,000$ randomly selected sentences (approx. 39K words) from the Brown corpus [FK82]. In figure 1 we present the accuracy after applying each rule. Although we are ultimately interested in the final accuracy, it is interesting to analyze the contribution of each rule to this accuracy. We would like to mention that in computing the accuracy, whenever our program cannot disambiguate a tag set, we score the

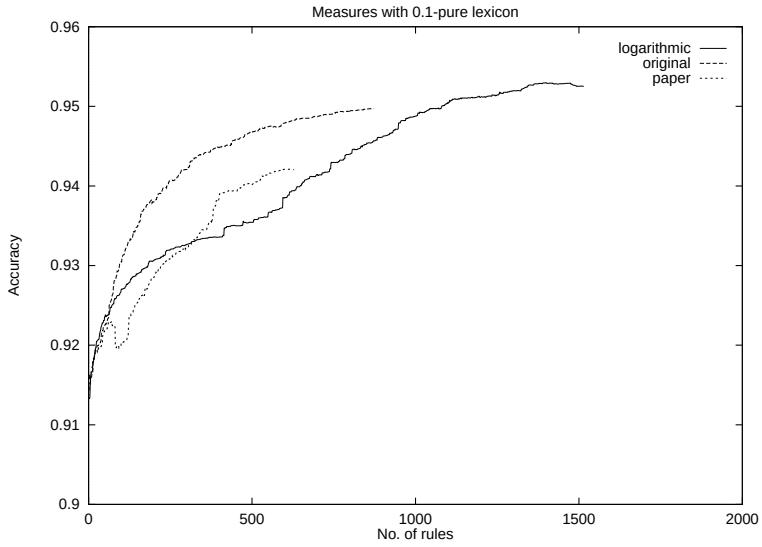

Figure 1: Behavior of different scoring functions

accuracy as if the tag would be chosen randomly. Therefore, if a tag set has two tags, one of which is correct, we only add 0.5 to the number of correctly tagged words.

The tests indicated that the purity of the lexicon has little influence on the relative accuracy of the measures we examined. For all purity values all the scoring functions had a similar behavior relative to each other. We will therefore limit our discussion to the lexicon "purity" value of 0.1. The final accuracies are summarized in table 1.

Scoring Function	Accuracy (%)	No. of rules
original	94.97	873
paper	94.21	628
logarithmic	95.25	1516

Table 1: Final accuracies for lexicon with purity 0.1

The highest accuracy is obtained by the logarithmic measure. This accuracy is however obtained at the expense of about twice as many rules, compared to the "original" measure.

The "paper" measure behaves the worst, seemingly because of choosing a wrong rule within the first couple hundred iterations. This rule causes a large drop in accuracy after which the tagger cannot catch up with the other scoring functions. The overall performance of the "paper" rule indicates however that variants of it might represent good scoring criteria.

Although the "original" measure does not reach the same accuracy as the logarithmic measure, it is more concise (converges in fewer rules). The small difference in accuracy does not justify the use of almost twice as many rules. The graph of the "original" scoring function is

monotonically increasing which is a very useful feature. We do not want to generate any rule that would decrease accuracy because the accuracy loss will have to be recovered by subsequent rules (as seen in the graph of the "paper" measure). These reasons justified the usage of the "original" measure in all subsequent tests. For all the other tests we have also used larger text sizes. Both training and test sets contained approximately $14,000$ randomly selected sentences (280K words) from the Brown corpus.

4.5 Influence of lexicon purity

In a sense, our code is not completely unsupervised, because it needs a lexicon relating words with possible tags. In order to create such a lexicon, one needs tagged text. It can be argued however, that the correspondence between words and tags does not vary too much among different texts. A data-set independent lexicon obtained from the Webster dictionary, for example, could be used. Such a lexicon is independent of the training text and therefore training can be considered unsupervised.

In order to avoid having to deal with unknown words, all tests reported in this paper have been run using an "exhaustive" lexicon containing all the words in both the training and the test set.

One problem that occurs when using a lexicon to initially tag a text, is that words might have been assigned too many tags. Some of those tags could appear because of typographical errors or because of rare meanings of the particular word. This makes disambiguation harder and decreases the accuracy of the tagger. Assigning each word its most frequent tag usually yields accuracies higher than 90%. The ambiguity introduced by having more than one tag per word however decreases the accuracy dramatically. Given that the tagger is unsupervised, it is impossible to prune the lexicon so that all wrong tags are avoided.

We chose the term "purity" to indicate the proportion of infrequent tags in the lexicon. For example a "purity" level of 0.2 means that for each word in the lexicon, if the most frequent tag for the word occurs n times, all tags that occur less than $0.2 \cdot n$ times do not appear in the lexicon. The name "purity" is not necessarily intuitive since a lexicon in which only the most frequent tags occur ("purity" 1) is definitely incorrect (i.e. there are tags which are not due to errors, which do not occur in the lexicon).

In figure 2 we show the results obtained by training the tagger with lexicons of different purity levels. The graph indicates the starting accuracy (the accuracy of initially tagging each word with all possible tags) and the ending accuracy (the accuracy obtained after applying all the rules) for each level of lexicon purity. The measure used during the tests was the "original" measure (see section 4.2).

The results show that the original accuracy increases for a while and then starts dropping. This behavior is natural since eliminating infrequent tags increases the accuracy of predicting the words with frequent tags. Uncommon words (or words which happen to be tagged with an uncommon tag for that particular word), will be incorrectly tagged, but their percentage is too small to make any difference in the overall accuracy. We gain more by more precisely tagging common words than we lose by incorrectly tagging uncommon words. After a point, however, the percentage of word–tag pairs assumed uncommon becomes large enough to matter in the global accuracy, and the overall accuracy starts decreasing.

To an extent, the same is true for the final accuracy. It is interesting to notice though, that

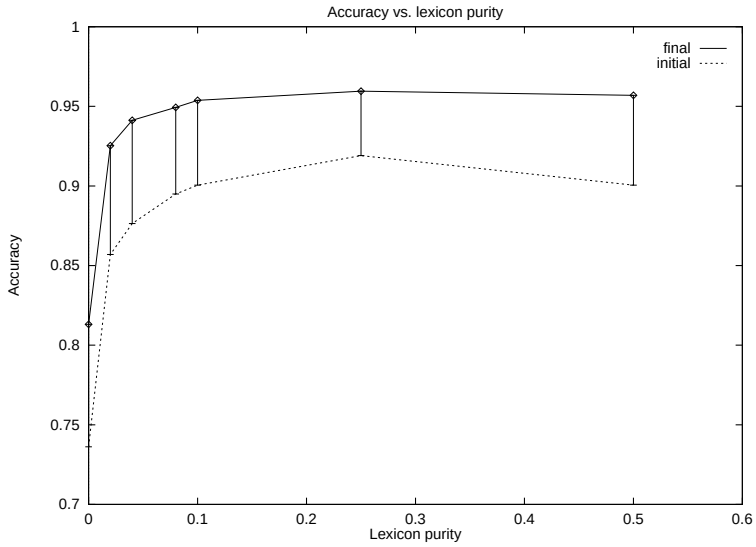

Figure 2: Influence of lexicon purity

the gain obtained by the tagger decreases with the increase of the initial accuracy. The increase in accuracy for the initial tagger is a result of reducing the initial ambiguity of frequent words. This reduction in ambiguity represents a restriction of the freedom the tagger has in order to choose the "right" tag, and therefore decreases the gain. When the accuracy of the initial tagger starts decreasing, the influence of the infrequent words becomes predominant. Due to the low frequencies of those words, their tags have very similar frequencies. This implies that the infrequent words have a good chance of remaining ambiguous as the lexicon "purity" increases. The effect of ambiguity combined with the increased influence of the uncommon words lead to a larger effect of the "smart" tagger. This influence is reflected in the increased accuracy gain and the slower decrease of the final accuracy.

Please note that increasing the "purity" of the lexicon, the two accuracies, initial and final, will converge to the accuracy of assigning each word the most probable tag. This portion of the graph is not shown in figure 2 since all interesting effects of lexicon "purity" occur for lexicons less than 0.5 "pure".

5 Comparison with HMM

Another approach to part-of-speech tagging was taken by Cutting, Kupiec, Pedersen, and Sibun [CKPS92]. They use a Hidden Markov Model in which states encode ambiguity classes, instead of tags. This approach introduced by Kupiec [Kup92] is known to have good results in practice. This part-of-speech tagger was developed at the Xerox PARC and the code is publicly available. We shall refer to it as the "Xerox tagger" or the "HMM tagger".

We ran tests to compare our tagger with the HMM tagger. We used a test and training

text of approximately 140,000 randomly selected sentences from the Brown corpus. We ran our tagger with the "original" measure. For the HMM tagger we used the parameters delivered with the Xerox distribution. During the tests we varied the purity of the lexicon. The results are summarized in figure 3.

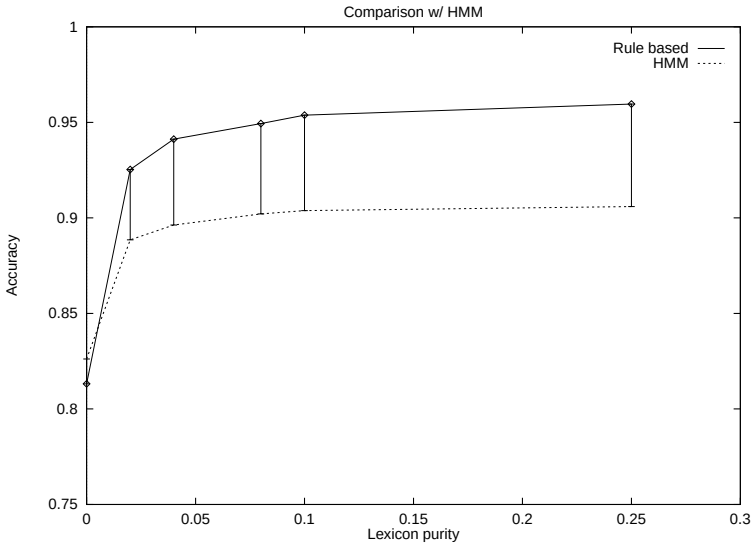

Figure 3: Comparison with HMM

Except for the case of the "dirty" lexicon (all tags have been retained), our code obtains higher accuracies than the HMM code. However the HMM tagger runs much faster (less than one hour for the HMM as opposed to tens of hours for the transformation-based tagger).

These results are surprising because the Xerox tagger has been reported to obtain an accuracy of 96% [CKPS92]. We believe that the poor performance may be caused by incorrect initialization of the transition probabilities. We have used the "out of the box" version of the tagger, without adapting the transition biases to our data set. We consider though that this is a fair comparison, first of all because the code we used is supposedly tailored for the Brown corpus. Secondly, the need to adjust parameters for different training data defeats the purpose of unsupervised learning.

6 Conclusions

The results we obtained show that transformation-based learning is a promising solution to unsupervised part-of-speech tagging. Although our code takes much longer time to train than the HMM code, we obtain much better accuracies without having to tweak any parameters. Also, the rules learned by our system are easy to understand and interpret. As an example, here are some of the rules determined by our tagger:

```
nn_vb vb LEFT in_nil_to
```

```
at_in_nn in RIGHT at
cs_in_nn_rb in RIGHT at
at_in_nn at RIGHT nn
cs_in_nn_rb in RIGHT at
```

The tags appearing in this example are: `nn` - noun, `vb` - verb, `in` - preposition or subordinating conjunction, `to` - to, `at` - article, `rb` - adverb, `cs` - comparison marker, and `nil` is the tag for any undetermined words. `LEFT` and `RIGHT` are context markers. First rule can thus be read as: **change tag set `nn_vb` into tag `vb` if the tag set to the left is `in_nil_to`**

The same cannot be said about a Hidden Markov Model where the parameters being learned are probabilities of state transitions. It is very hard to associate meaning to the parameters determined while training an HMM.

In the current implementation the time needed by our tagger to tag text is much longer than the time needed by the Xerox tagger. This difference is however not that significant because the set of rules can be transformed into a Finite-State Transducer as described in [RS95]. This way the tagging times of the two systems would be similar, the only major differences being in the training time.

In the end we would like to mention that the code of the part-of-speech unupervised tagger is available from `ftp://ftp.cs.jhu.edu/pub/brill/Programs/UNSUP_TAGGER_V0.5.tar.Z`

7 Future work

During the work mentioned in this paper we have uncovered several continuation paths which we haven't yet explored. This section is meant to describe possible ideas for the further development of unsupervised part-of-speech tagging.

"unsupervised" lexicon - as we previously mentioned, during our tests we used a lexicon derived from the training and the test data. This approach violates the assumption of "unsupervised training". Tests need to be performed using a generic lexicon such as one obtained from the Webster dictionary.

automatic lexicon pruning - our tests show that correctly pruning the lexicon may substantially increase the accuracy of the tagger. We tried a "bootstrapping" approach in order to prune the lexicon in an unsupervised manner. The text was first tagged using a "dirty" lexicon (where all tags are present and no frequency information is available). After that, the lexicon was pruned based on frequency information determined from the tagged text. After a few repetitions of this procedure we noticed improvement in the final accuracy of the tagger. The accuracy does not reach the levels achieved by manually selecting the lexicon "purity" but we believe that variations of this procedure should achieve very good results.

unknown words - during all the tests reported in this paper we avoided the use of unknown words (words that do not appear in the lexicon). We did however run several tests in which unknown words were uniformly assigned a set of possible tags. Due to the small number of unknown words (usually less than 10%) and contextual information, the tagger is able to obtain accuracies of up to 80% on the unknown words. Further tests should be done. The method for assigning initial tags to unknown words could also be improved.

generalizing rules - some of the rules derived by the tagger can be generalized. For example, it can happen that in a certain context, rules pertinent to that context will all resolve to the same tag T. It might make sense to replace all those rules with a "general" rule mentioning that in that particular context, any tag set should resolve to T.

order of applying the rules - at the moment we apply the rules in the order in which we discovered them. There is however no reason to believe that this is the only reasonable strategy. Alternative strategies should therefore be explored.

context size - the part-of-speech tagger presented in this paper only looks at the previous or next word/tag as a context for the current word. It is interesting to explore if improvements can be achieved by considering more than one token surrounding the current word.

scoring functions - we have presented three scoring functions we have tested. Other scoring functions could be constructed such that they either achieve better accuracies or converge in fewer steps.

References

[Bri95] Eric Brill. Unsupervised learning of disambiguation rules for part of speech tagging. *3rd Workshop on Very Large Corpora*, 1995.

[CKPS92] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992. ACL.

[FK82] W. Francis and H. Kučera. *Frequency analysis of English usage: Lexicon and grammar*. Houghton Mifflin, Boston, 1982.

[Kup92] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, 6:225–242, 1992.

[Lee89] Kai-Fu Lee. *Automatic Speech Recognition*. Kluwer Academic Publishers, 1989.

[RS95] Emmanuel Roche and Yves Shabes. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 21(2):227–253, 1995.

[Yar95] David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. *Proceedings of ACL*, 1995.