

Parameter reduction in unsupervisedly trained sliding-window part-of-speech taggers

Enrique Sánchez-Villamil and Mikel L. Forcada and Rafael C. Carrasco

{esvillamil,mlf,carrasco}@dlsi.ua.es

Transducens, Departament de Llenguatges i Sistemes Informàtics

Universitat d'Alacant, E-03071 Alacant, Spain

Abstract

A new, robust sliding-window part-of-speech tagger is presented, which itself is an approximation of an existing model, and a method is described to estimate its parameters from an untagged corpus. The approximation reduces the memory requirements without a significant loss in accuracy. Its performance is compared to that of the original sliding-window tagger as well as to that of a standard Baum-Welch-trained hidden-Markov-model part-of-speech tagger and a random tagger.

1 Introduction

A large fraction (typically 30%, but varying from one language to another) of the words in natural language texts are words that, in isolation, may be assigned more than one morphological analysis and, in particular, more than one part of speech (PoS). The correct resolution of PoS ambiguity for each occurrence of the word in the text is crucial in many natural language processing applications; for example, in machine translation, the correct equivalent of a word may be very different depending on its PoS.

This paper presents a new version of a sliding-window (SW) PoS tagger, that is, a system which assigns the PoS of a word based on the information provided by a fixed window of words around it. The SW tagger idea is not new (Sánchez-Villamil *et al.* 04), but the number of parameters required to achieve acceptable results is high compared to that of more usual approaches such as hidden Markov Models (HMM). The new light sliding-window (LSW) PoS tagger proposed here reduces greatly the number of parameters with a negligible loss of performance.

The paper is organized as follows: section 2 gives some definitions and describes the notation that will be used throughout the paper; section 3 describes the approximations that allow a SW tagger to be trained in an unsupervised manner and the training process itself; section 4 describes the LSW tagger in parallel to

the SW tagger training algorithm; section 5 describes a series of experiments performed to compare the performance of a LSW tagger to that of a HMM tagger and to that of the SW tagger; and, finally, concluding remarks are given in section 6.

2 Preliminaries

Let $\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_{|\Gamma|}\}$ be the *tagset* for the task, that is, the set of PoS tags a word may receive in a specific language, and $W = \{w_1, w_2, \dots, w_{|W|}\}$ be the *vocabulary* of the task. A partition of W is established so that $w_i \equiv w_j$ (that is, both words belong to the same equivalence class) if and only if both are assigned the same subset of tags by the lexical categorizer.¹

It is usual (Cutting *et al.* 92) to refine this partition so that, for high-frequency words, each word class contains just one word whereas, for lower-frequency words, word classes are made to correspond exactly to *ambiguity classes* containing all words receiving the same subset of PoS tags (although it would also be possible to use one-word classes for all words or to use only ambiguity classes). This refinement allows for improved performance on very frequent ambiguous words while keeping the number of parameters of the tagger under control.

Any such refinement will be denoted as $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{|\Sigma|}\}$ where σ_i are word classes. In this paper, word classes will simply be ambiguity classes, without any refinement. We will call $T : \Sigma \rightarrow 2^\Gamma$ the function returning the set $T(\sigma)$ of PoS tags for each word class σ .

The PoS tagging problem may be formulated as follows: given a text $w[1]w[2] \dots w[L] \in W^+$, each word $w[t]$ is assigned a word class $\sigma[t] \in \Sigma$ to obtain

¹The lexical categorizer function may be implemented by a dictionary, a morphological analyser, a guesser, or any combination thereof.

an *ambiguously tagged* text $\sigma[1]\sigma[2]\dots\sigma[L] \in \Sigma^+$; the task of the PoS tagger is to obtain a *tagged* text $\gamma[1]\gamma[2]\dots\gamma[L] \in \Gamma^+$ (with all $\gamma[t] \in T(\sigma[t])$) as correct as possible.

Statistical PoS tagging looks for the *most likely* tagging $\gamma^*[1], \gamma^*[2], \dots, \gamma^*[L]$ given an ambiguously tagged text $\sigma[1]\sigma[2]\dots\sigma[L]$:

$$\begin{aligned} \gamma^*[1]\dots\gamma^*[L] = \\ \operatorname{argmax}_{\gamma[t] \in T(\sigma[t])} P(\gamma[1]\dots\gamma[L] \mid \sigma[1]\dots\sigma[L]). \end{aligned} \quad (1)$$

ambiguously tagged sequence $\sigma[1]\dots\sigma[L]$. In hidden Markov models (Rabiner 89), use of the Bayes' formula, modelling of tag sequences as first-order Markov processes, and additional approximations lead to

$$\begin{aligned} \gamma^*[1]\dots\gamma^*[L] = \\ \operatorname{argmax}_{\gamma[t] \in T(\sigma[t])} \prod_{t=0}^{t=L} p_S(\gamma[t+1] \mid \gamma[t]) \times \\ \prod_{t=1}^{t=L} p_L(\sigma[t] \mid \gamma[t]), \end{aligned} \quad (2)$$

where P_S is the *syntactical* probability modelling tag sequences and P_L is the *lexical* probability modelling the relations between tags and word classes, with $\gamma[0] = \gamma[L+1] = \gamma_{\#}$, a special delimiting tag analogous to a sentence boundary. The number of trainable parameters is $(|\Gamma| + |\Sigma|)|\Gamma|$. Tagging (searching for the optimal $\gamma^*[1]\gamma^*[2]\dots\gamma^*[L]$) is implemented using an efficient, left-to-right algorithm usually known as Viterbi's algorithm (Cutting *et al.* 92; Rabiner 89), which, if conveniently implemented, can output a partial tagging each time a nonambiguous word is seen, but has to maintain multiple hypotheses when reading ambiguous words. HMM taggers may be trained either from tagged text (simply by counting and taking probabilities to be equal to frequencies) or from untagged text, using the well-known expectation-maximization backward-forward Baum-Welch algorithm (Rabiner 89; Cutting *et al.* 92).

3 The Sliding-Window PoS Tagger model

The sliding-window PoS tagger (Sánchez-Villamil *et al.* 04) approximates the probability in eq. (1) *directly* as follows:

$$P(\gamma[1]\gamma[2]\dots\gamma[L] \mid \sigma[1]\sigma[2]\dots\sigma[L]) \simeq \prod_{t=1}^{t=L} p(\gamma[t] \mid C_{(-)}[t]\sigma[t]C_{(+)}[t]) \quad (3)$$

where $C_{(-)}[t] = \sigma[t-N_{(-)}]\sigma[t-N_{(-)}+1]\dots\sigma[t-1]$ is a *left context* of word classes of length $N_{(-)}$ and $C_{(+)}[t] = \sigma[t+1]\sigma[t+2]\dots\sigma[t+N_{(+)}]$ is a *right context* of word classes of length $N_{(+)}$, so that for $t < 1$ and $t > L$, $\sigma[t] = \sigma_{\#}$, a special delimiting word class such that $T(\sigma_{\#}) = \{\gamma_{\#}\}$.

This *sliding window* method is local in nature; it does not consider any context beyond the window of $N_{(-)}+N_{(+)}+1$ words; its implementation is straightforward, even more than that of Viterbi's algorithm. The main problem is the estimation of the probabilities $p(\gamma[t] \mid C_{(-)}[t]\sigma[t]C_{(+)}[t])$. If a tagged corpus is available, these probabilities may be easily obtained by counting; however, the SW tagger has a specific way of estimating them from an untagged corpus, as we will see below. Another problem is the large number of parameters of the model $(|\Sigma|^{N_{(+)}+N_{(-)}}|\Gamma|)$.

The main approximation in the model consists in assuming that the best tag $\gamma^*[t]$ contained in the window depends on the preceding context $C_{(-)}[t]$ and the succeeding context $C_{(+)}[t]$, and only *selectionally* on the word (one could say that it is the context which determines the probabilities of each tag, whereas the word just *selects* tags among those in $T(\sigma[t])$).

The most probable tag $\gamma^*[t]$ is

$$\gamma^*[t] = \operatorname{argmax}_{\gamma \in T(\sigma[t])} p(\gamma[t] = \gamma \mid C_{(-)}[t]\sigma[t]C_{(+)}[t]). \quad (4)$$

We will drop the position index $[t]$ because of time invariance; and write $p(\gamma \mid C_{(-)}\sigma C_{(+)})$. These probabilities are easily estimated from a tagged corpus (e.g., by counting) but estimating them from an untagged corpus involves an iterative process, which proceeds by estimating counts $\tilde{n}_{C_{(-)}\gamma C_{(+)}}$ which express the *effective* number of times that tag γ would appear in the text between contexts $C_{(-)}$ and $C_{(+)}$. Therefore,

$$p(\gamma \mid C_{(-)}\sigma C_{(+)}) = k_{C_{(-)}\sigma C_{(+)}} \tilde{n}_{C_{(-)}\gamma C_{(+)}} \quad (5)$$

if $\gamma \in T(\sigma)$ and zero otherwise, where $k_{C_{(-)}\sigma C_{(+)}} = (\sum_{\gamma' \in T(\sigma)} \tilde{n}_{C_{(-)}\gamma' C_{(+)}})^{-1}$ is a normalization factor. Accordingly, equation (4) could be written as:

$$\gamma^*[t] = \operatorname{argmax}_{\gamma \in T(\sigma[t])} \tilde{n}_{C_{(-)}[t]\gamma C_{(+)}[t]}, \quad (6)$$

where the dependence with respect to $\sigma[t]$ can be clearly seen to be only selectional.

But, how can the counts $\tilde{n}_{C_{(-)}\gamma C_{(+)}}$ be estimated? If the window probabilities $p(\gamma \mid C_{(-)}\sigma C_{(+)})$ were known, the effective counts could be easily obtained

from the text itself as follows:

$$\tilde{n}_{C_{(-)}\gamma C_{(+)}} = \sum_{\sigma:\gamma\in T(\sigma)} n_{C_{(-)}\sigma C_{(+)}} p(\gamma | C_{(-)}\sigma C_{(+)}) , \quad (7)$$

where $n_{C_{(-)}\sigma C_{(+)}}$ is the number of times that ambiguity class σ appears between contexts $C_{(-)}$ and $C_{(+)}$; that is, one would add $p(\gamma | C_{(-)}\sigma C_{(+)})$ each time a word class σ containing tag γ appears between $C_{(-)}$ and $C_{(+)}$. Equations (5) and (7) may be iteratively solved until the $\tilde{n}_{C_{(-)}\gamma C_{(+)}}$ converge. For the computation to be more efficient, one can avoid storing the probabilities $p(\gamma | C_{(-)}\sigma C_{(+)})$ by organizing the iterations around the $\tilde{n}_{C_{(-)}\gamma C_{(+)}}$ as follows, by combining eqs. (5) and (7) and using an iteration index denoted with a superscript $[m]$,

$$\tilde{n}_{C_{(-)}\gamma C_{(+)}}^{[m]} = \tilde{n}_{C_{(-)}\gamma C_{(+)}}^{[m-1]} \times \sum_{\sigma:\gamma\in T(\sigma)} n_{C_{(-)}\sigma C_{(+)}} \left(\sum_{\gamma'\in T(\sigma)} \tilde{n}_{C_{(-)}\gamma' C_{(+)}}^{[m-1]} \right)^{-1} , \quad (8)$$

where the iteration may be easily seen as a process of successive multiplicative corrections to the effective counts $\tilde{n}_{C_{(-)}\gamma C_{(+)}}$. A convenient starting point is given by $p(\gamma | C_{(-)}\sigma C_{(+)}) = |T(\sigma)|^{-1}$ which is equivalent to assuming that initially all possible tags are equally probable for each word class.

Equation (8) contains the counts $n_{C_{(-)}\sigma C_{(+)}}$ which depend on $N_{(+)} + N_{(-)} + 1$ word classes; if memory is at a premium, instead of reading the text once to count these and then iterating, the text may be read in each iteration to avoid storing the $n_{C_{(-)}\sigma C_{(+)}}$, and the $\tilde{n}_{C_{(-)}\gamma C_{(+)}}^{[k]}$ may be computed *on the fly*. Iterations proceed until a selected convergence condition has been met (e.g. a comparison of the $\tilde{n}_{C_{(-)}\gamma C_{(+)}}^{[k]}$ with respect to the $\tilde{n}_{C_{(-)}\gamma C_{(+)}}^{[k-1]}$, or the completion of a predetermined number of iterations).

4 Light Sliding-Window PoS Tagger model

The model proposed in this paper may be considered as an approximation to the SW tagger just described, with the objective of reducing the number of parameters to estimate without a significant loss in tagging accuracy. The number of parameters of the LSW tagger in the worst case is $|\Gamma|^{N_{(-)}+N_{(+) + 1}$, compared to the $|\Sigma|^{N_{(-)}+N_{(+) + 1} |\Gamma|$ of the SW tagger. The number of parameters of the LSW tagger depends only on the size of the set of tags, which is much smaller than the number of word classes. However, as expected, the reduction of parameters makes the tagger slower, as the

training and tagging equations are more complicated to compute.

The best tag γ^* is obtained by considering for each possible $\gamma[t]$ *all possible disambiguations* $E_{(-)}[t]\gamma[t]E_{(+)}[t]$ of the current window $C_{(-)}[t]\sigma C_{(+)}[t]$ and adding their probabilities $p(E_{(-)}[t]\gamma[t]E_{(+)}[t] | C_{(-)}[t]\sigma[t]C_{(+)}[t])$ as if they were independent.

The LSW tagger approximates eq. (4) as follows:

$$\gamma^*[t] = \operatorname{argmax}_{\gamma\in T(\sigma[t])} \sum_{\substack{E_{(-)}\in T'(C_{(-)}[t]) \\ E_{(+)}\in T'(C_{(+)}[t])}} p(E_{(-)}\gamma E_{(+)} | C_{(-)}[t]\sigma[t]C_{(+)}[t]) \quad (9)$$

where $E_{(-)}[t] = \gamma[t - N_{(-)}]\gamma[t - N_{(-)} + 1] \dots \gamma[t - 1]$ is a *left context of tags* of size $N_{(-)}$, $[E_{(+)}[t] = \gamma[t + 1]\gamma[t + 2] \dots \gamma[t + N_{(+)}]$ is a *right context of tags* of size $N_{(+)}$, and $\gamma[t] \forall t < 1, \forall t > L$ are all set to the special delimiting tag $\gamma_{\#}$.

Let $T' : \Sigma^* \rightarrow 2^{\Gamma^*}$ now be the function that returns all the tag sequences that can be assigned to a given sequence of ambiguity classes. The probabilities $p_{E_{(-)}\gamma E_{(+)}}$ may be easily estimated in an analogous way to equation (5), dropping time indices for invariance:

$$p(E_{(-)}\gamma E_{(+)} | C_{(-)}\sigma C_{(+)}) = k_{C_{(-)}\sigma C_{(+)}} \tilde{n}_{E_{(-)}\gamma E_{(+)}} \quad (10)$$

if $E_{(-)}\gamma E_{(+)} \in T'(C_{(-)}\sigma C_{(+)})$, and zero otherwise, where $k_{C_{(-)}\sigma C_{(+)}} = \left(\sum_{\gamma\in T(\sigma), E_{(-)}\in T'(C_{(-)}[t]), E_{(+)}\in T'(C_{(+)}[t])} \tilde{n}_{E_{(-)}\gamma E_{(+)}} \right)^{-1}$ is a normalization factor. Thus, equation (9) could be written similarly to eq. (6) as:

$$\gamma^*[t] = \operatorname{argmax}_{\gamma\in T(\sigma[t])} \sum_{\substack{\sigma:\gamma\in T(\sigma) \\ C_{(-)}:E_{(-)}\in T'(C_{(-)}[t]) \\ C_{(+)}:E_{(+)}\in T'(C_{(+)}[t])}} \tilde{n}_{E_{(-)}\gamma E_{(+)}} ; \quad (11)$$

as in (6), $\gamma^*[t]$ depends only selectionally on $\sigma[t]$.

The counts $\tilde{n}_{E_{(-)}\gamma E_{(+)}}$ could be easily estimated if the probabilities $p(E_{(-)}\gamma E_{(+)} | C_{(-)}\sigma C_{(+)})$ were known, using an equation parallel to (7):

$$\tilde{n}_{E_{(-)}\gamma E_{(+)}} = \sum_{\substack{\sigma:\gamma\in T(\sigma) \\ C_{(-)}:E_{(-)}\in T'(C_{(-)}) \\ C_{(+)}:E_{(+)}\in T'(C_{(+)})}} \left(n_{C_{(-)}\sigma C_{(+)}} \times p(E_{(-)}\gamma E_{(+)} | C_{(-)}\sigma C_{(+)}) \right) \quad (12)$$

But, since they are unknown, the counts are estimated through an adapted version of the iterative

equation (8) of the SW tagger, applied until it converges:

$$\tilde{n}_{E_{(-)}\gamma E_{(+)}}^{[m]} = \tilde{n}_{E_{(-)}\gamma E_{(+)}}^{[m-1]} \times \sum_{\substack{\sigma:\gamma\in T(\sigma) \\ C_{(-)}:E_{(-)}\in T'(C_{(-)}) \\ C_{(+)}:E_{(+)}\in T'(C_{(+)})}} n_{C_{(-)}\sigma C_{(+)}} k_{C_{(-)}\sigma C_{(+)}}^{[m-1]} \quad (13)$$

A convenient equiprobable initialization takes $p(E_{(-)}\gamma E_{(+)} | C_{(-)}\gamma C_{(+)}) = (|T'(C_{(-)}\sigma C_{(+)})|)^{-1}$.

As has been advanced, the main difference between the SW and LSW models is the number of parameters needed; while the SW tagger keeps all $\tilde{n}_{C_{(-)}\gamma C_{(+)}}$, the LSW tagger keeps only the $\tilde{n}_{E_{(-)}\gamma E_{(+)}}$; this results in a lower complexity in the worst case at the expense of an increase in tagging time, given that the computation of $\gamma^*[t]$ needs to consider $|\Gamma|^{N_{(-)}+N_{(+)}}$ effective counts instead of only one, as in the original. Moreover, it is clear that a reduction in the number of parameters means a loss of information, so the tagging accuracy is expected to be worse.

5 Experiments

This section reports experiments to assess the performance of the sliding-window PoS taggers using different amounts of context, and compares them with that of customary Baum-Welch-trained HMM taggers (Cutting *et al.* 92).

For training and testing we have used the Penn Treebank, version 3 (Marcus *et al.* 93; Marcus *et al.* 94), which has 1,014,377 PoS-tagged words of English text taken from *The Wall Street Journal*. The word classes Σ of the Treebank will be taken simply to be ambiguity classes. The Treebank uses 45 different PoS tags; 24.08% of the words are ambiguous.

The experiments use a lexicon extracted from the Penn Treebank, that is, a list of words with all the possible parts of speech observed.² Of course, the exact tag given in the Treebank for each occurrence of each word is taken into account only for testing but not for training. To simulate the effect of using a real, limited lexical categorizer, we have filtered the resulting lexicon to keep only the 14,276 most frequent words (95% text coverage), and to remove, for each word, any PoS tag occurring less than 5% of the time. Using this simplified, but realistic, lexicon, texts in the Penn Treebank show 218 ambiguity classes (the word classes for these experiments). Words not included in the lexicon are assigned to a special ambiguity class

²Even if the Treebank were ambiguously tagged (i.e. with ambiguity classes), a lexicon could still be extracted.

(the *open* class) containing all tags representing parts of speech that can grow (i.e. a new word can be a noun or a verb but hardly ever a preposition).³

In order to train the taggers we have applied the following strategy, so that we can use as much text as possible for training: the Treebank is divided into 20 similarly-sized sections; a leaving-one-out procedure is applied, using 19 sections for training and the remaining one for testing, so that our results are the average of all 20 different train–test configurations. In our experiments, the SW model was a 15% faster in training time than LSW, but in our implementation *there was no significant difference in tagging time* between the models. Both models tag around 70,000 words per second in a Pentium IV 2.8GHz.

5.1 Effect of the amount of context

First of all, we show the results of the sliding-window taggers using no context ($N_{(-)} = N_{(+)} = 0$) as a baseline, and compare them to those of a Baum-Welch-trained HMM tagger and to random tagging. As expected, the performance of the taggers without context is not much better than random tagging (see table 1). This happens because without context the SW tagger and the LSW tagger, whose behaviours are completely equivalent in this case, simply deliver an estimate of the most likely tag in each class. The HMM tagger accuracy (90.7%) is also given for comparison. In this and the rest of experiments reported here, standard deviations are in the range 0.25% – 0.30%, but they will not be shown for clarity. All results correspond to the 15th iteration of the SW, LSW and HMM models, although the SW and LSW taggers usually converge in 3 or 4 iterations.

In order to improve the results, one obviously needs to increase the context (i.e., widen the sliding window). The results of using a reduced context of only one word before the current word ($N_{(-)} = 1, N_{(+)} = 0$) (the results obtained using a context of one word *after* the current word are worse) are also shown in figure 1. It is worth noting that even using such a limited context the performance of the LSW tagger almost reaches that of the HMM tagger, and is comparable —within the standard deviation— to that of the SW tagger, which has five times more parameters.

If we increase the size of the context to two context words, we have three different possibilities: using the two immediately preceding words, using one preceding and one succeeding word, and using two succeeding

³Our open class contains the Penn Treebank tags CD, JJ, JJR, JJS, NN, NNP, NNPS, RB, RBR, RBS, UH, VB, VBD, VBG, VBN, VBP, and VBZ.

Tagger	$N_{(-)}$	$N_{(+)}$	Number of parameters	Accuracy
RANDOM	-	-	0	85.0%
HMM	-	-	11,835	90.7%
LSW AND SW	0	0	45	86.4%
SW	1	0	9,810	90.4%
LSW	1	0	2,025	90.2%
SW	1	1	2,138,580	92.1%
LSW	1	1	91,125	91.8%

Table 1: Comparison of the accuracy and the number of parameters of sliding-window taggers to other tagging strategies, as a function of the size of the left and right contexts.

ing words; the best results are achieved when using one preceding and one succeeding word ($N_{(-)} = 1$ and $N_{(+)} = 1$), and are shown in table 1. The performance of the sliding-window taggers is now clearly better than that of the HMM tagger, in exchange for a large increase in the number of parameters (still moderate in the case of the LSW tagger). Increasing the context a bit more, until using three context words in all possible geometries does not improve results (the corpus is not large enough to allow the estimation of so many parameters).

5.2 Effect of corpus size

To assess the effect of corpus size, we trained the taggers with corpora built using an increasing number of sections of the Treebank. The results show that the LSW tagger reaches its peak performance with smaller corpora than the SW tagger, which was expected in view of the difference in the number of parameters.

6 Concluding remarks

As commonly-used HMM taggers, simple and intuitive sliding-window PoS taggers (SW taggers, (Sánchez-Villamil *et al.* 04)) may be iteratively trained in an unsupervised manner using reasonable approximations to reduce the number of trainable parameters (LSW taggers, proposed here). Experimental results show that the performance of the sliding-window taggers and HMM taggers having a similar number of trainable parameters is comparable; the best results are obtained with a context of one preceding and one succeeding word. LSW tagger results are almost indistinguishable from SW tagger results. Besides, the reduction of parameters allows the LSW tagger to be trained with a smaller training set.

We are currently studying ways to improve the training algorithm, so that incremental training (i.e. automatically adding new text to the training set)

can be done. We also plan to test the models with different corpora, using the morphological analysers and finer tagsets in the Spanish–Catalan translator `interNOSTRUM.com` (Canals-Marote *et al.* 01). In addition, we are studying the introduction of constraints (Laporte & Monceaux 00) and lexicalization (using word classes finer than ambiguity classes).

Acknowledgements: Work funded by the Spanish Government through grant TIC2003-08681-C02-01.

References

- (Canals-Marote *et al.* 01) R. Canals-Marote, A. Esteve-Guillen, A. Garrido-Alenda, M. Guardiola-Savall, A. Iturraspe-Bellver, S. Montserrat-Buendia, S. Ortiz-Rojas, H. Pastor-Pina, P.M. Pérez-Antón, and M.L. Forcada. The Spanish-Catalan machine translation system `interNOSTRUM`. In B. Maegaard, editor, *Proceedings of MT Summit VIII: Machine Translation in the Information Age*, pages 73–76, 2001. Santiago de Compostela, Spain, 18–22 July 2001.
- (Cutting *et al.* 92) D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing. Association for Computational Linguistics. Proceedings of the Conference*, pages 133–140, Trento, Italia, 31 marzo–3 abril 1992.
- (Laporte & Monceaux 00) É. Laporte and A. Monceaux. Elimination of lexical ambiguities by grammars: The ELAG system. In John Benjamins Publishing Company, editor, *Linguisticae Investigationes*, volume 22, pages 341–367(27), 2000.
- (Marcus *et al.* 93) Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: the Penn Treebank. *Computational linguistics*, 19:313–330, 1993. Reprinted in Susan Armstrong, ed. 1994, *Using large corpora*, Cambridge, MA: MIT Press, 273–290.
- (Marcus *et al.* 94) Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating predicate argument structure. In *Proc. ARPA Human Language Technology Workshop*, pages 110–115, 1994.
- (Rabiner 89) Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- (Sánchez-Villamil *et al.* 04) E. Sánchez-Villamil, Mikel L. Forcada, and Rafael C. Carrasco. Unsupervised training of a finite-state sliding-window part-of-speech tagger. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence*, 3230(12):454–463, 2004.