Combining POS-taggers for improved accuracy on Swedish text

Jonas Sjöbergh
KTH Nada
Stockholm, Sweden
jsh@nada.kth.se

Abstract

Several POS-taggers are trained and tested on Swedish text. Methods to improve the accuracy of the tagging are then examined. These methods include voting, letting taggers change their voting contribution depending on how confident they are and training a new second level classifier on the output of the taggers. All these methods are more accurate than the most accurate original tagger, with 15% less errors or better.

Which types of errors these methods correct and which types remain are also examined. The number of errors in some common error categories actually increase, while many uncommon errors are corrected.

1 Introduction

Part-of-speech tagging is a very important step in most advanced language technology systems. It is a nontrivial problem due to ambiguous words and unknown words, i.e. words not in the lexicon. POS-tagging is harder for some languages than others. Typical accuracy for Swedish taggers is between 94% and 96% (Megyesi, 2001). Taggers may either be based on manually written rules for a specific language (Karlsson et al., 1995), language independent, but trained on a tagged corpus (Brants, 2000; Daelemans et al., 2001; Ratnaparkhi, 1996; Schmid, 1994; Brill, 1992) or a combination of both (Carlberger and Kann, 1999).

A straightforward way to improve tagging is to combine the results of several taggers, hoping to take advantage of the fact that different taggers are good at tagging different constructions. The more general problem of combining different classifiers, in this case taggers, also known as using ensembles of classifiers, has been tried before. A good overview of why ensembles are a good idea and different ways of constructing and combining classifiers is given in (Dietterich, 1997). The basic idea is that classifiers making uncorrelated errors can correct each other.

The error reduction achieved by combining classifiers has been shown to be negatively correlated with how correlated the errors made by the classifiers are (Ali and Pazzani, 1996). Classifiers that are different from each other in some way are likely to make uncorrelated errors, thus different ways of creating a diverse classifier ensemble have been studied. These include using different classifier algorithms, using different training sets, using different data features (or feature weights) and generating different pseudo-examples for training, see for instance (Tumer and Ghosh, 1996) for examples from classifying in general or (Màrquez et al., 1999) for POS-tagging examples.

Common ways of combining POS-taggers include voting, possibly weighted, training a new classifier on the output of the taggers and hand written rules choosing a tagger based on for instance text type or linguistic context, see for instance (Brill and Wu, 1998), (van Halteren et al., 1998) and (Borin, 2000).

Combining classifiers in the context of POS-tagging has mainly been used to increase tagging accuracy. Other uses include automatically creating a larger training corpus by bootstrapping and combining two taggers (Màrquez et al., 1998) and using an ensemble of taggers to filter out synthetic noise (deliberately added tagging errors) in a pre-tagged corpus (Berthelsen and Megyesi, 2000).

We have trained and evaluated seven taggers on Swedish text. Specifically we have tried to maximize the accuracy by combining the taggers in different ways.

1.1 Training and evaluation

The Stockholm-Umeå Corpus (SUC) (Ejerhed et al., 1992), a manually corrected tagged corpus of Swedish, was used for training and testing. The tag set in SUC was slightly modified, resulting in a tag set of 150 tags.

Training and testing was performed by splitting SUC into two parts: a test set consisting of about 60 000 words, and a training set consisting of the rest of the corpus, about 1.1 million words. This results in approximately 5% of the words in the test data being unknown words (words not in the training data). To increase reliability of results, the part of SUC used as test data was chosen in 10 different ways (with all 10 test sets being disjoint) and the training and testing repeated once for every choice. The test data was chosen to be as balanced as possible.

Testing was done by stripping the tags from the test data and letting the taggers tag the text. The assigned tags were then compared to the original tags of the test data and if the assigned tag for a word was the same as the original tag it was deemed correct, otherwise incorrect. Data was gathered using AutoEval (Bigert et al., 2003). This method results in some unfairness, as SUC (and thus the test and training data) contains some erroneous taggings and some inconsistent taggings. Recently, some of the tests were run again on a newer version of SUC, where some taggings have been changed (presumably corrected). This gave slight improvements on all tested taggers, one typical example is an increase in accuracy from 95.9% to 96.0%. Also, in some cases several tags could be seen as correct, but only the one chosen in SUC would be counted as such. A thorough discussion of evaluation of automatic taggers, tagging errors and ambiguous words in SUC can be found in (Källgren, 1996).

1.2 Tested taggers

The taggers to use were chosen by finding taggers that were easily available, language independent enough to be used on Swedish text and easy to train on new data. All tested taggers were run with their default options. The following taggers were tested:

- fnTBL (Ngai and Florian, 2001), a transformation based tagger.
- Granska (Carlberger and Kann, 1999), a trigram HMM-tagger.
- Mxpost (Ratnaparkhi, 1996), a maximum entropy tagger.
- Stomp (Sjöbergh, 2003)[1], a tagger that matches word sequences between training and test data.
- Timbl (Daelemans et al., 2001), a memory based tagger. [2]

  Timbl was also trained as a second level classifier to combine results of other taggers.
- TnT (Brants, 2000), a trigram HMM-tagger.
- TreeTagger (Schmid, 1994), a tagger using decision trees.

[1]In this paper an old version of Stomp was used. A newer version, much faster and with higher tagging accuracy is described in the cited paper. The old version was the one available at the time of these tests, and it works better in ensembles, despite being less accurate on its own. The difference between the two versions is the handling of unknown words, the old version uses only the context of an unknown word where the new version also uses the suffix of the word.

[2]Timbl can use either decision trees or memory based learning. Only memory based learning was used in these experiments.

Timbl has no "default" option for POS-tagging, so features had to be selected. It was trained on the following features: for known words: the word itself, the two preceding assigned tags, ambiguity class (possible tags for word), ambiguity class of next word; for unknown words: the two preceding assigned tags, last 4 letters (each a different feature), word is capitalized flag, ambiguity class of next word. All words in any of the open word classes were used for training the classification of unknown words. Better features could probably be selected, making Timbl more accurate, but this was deemed accurate enough.

Tagger	Accuracy (%) (all words)	Accuracy (%) (unknown words)	Training time	Tagging time
Baseline[1]	87.3	25.4	34 s	13 s
fnTBL	95.6	79.8	2 h[2]	2 min[2]
Granska Original[3]	96.0	89.5	6 min	41 s
Granska[3]	95.4	88.4	6 min	41 s
Mxpost	95.5	85.1	13 h	4 min
Stomp	93.8	63.3	0	2.5 min
Timbl	94.7	79.1	8.5 min	1 h
TnT	95.9	88.5	20 s	8 s
TreeTagger	95.1	77.5	35 s	5 s

[1] A unigram tagger: choose most common tag for known words and choose most common open word class tag for unknown words. This tagger was not used in later experiments, just for comparison here.

[2] The SunBlade otherwise used did not have enough memory to run fnTBL, so a Pentium III 1100 MHz (about twice as fast) was used instead.

[3] Granska normally tokenizes text differently than the text in the test data, mainly by combining some constructions of several words into one token. This makes it hard to use in an ensemble, so another version of Granska, which keeps the original tokenization, was used whenever Granska was used in an ensemble. This version is quite a bit worse than the original. The accuracy of the original version is shown for comparison with the accuracy of the ensemble methods (since it happened to be the most accurate tagger). The other, less accurate, version was used everywhere else.

Table 1: Tagging accuracy on Swedish text. Measurements are for training data of 1.1 million words, 5% of words in the test data were unknown words. Training and tagging time was measured on a SunBlade 100.

Tagger	Accuracy (all)	Accuracy (unknown words)
Best tagger	96.0	89.5
Best voting	96.6	90.2

Table 2: Accuracy of the best single tagger and of voting taggers for the best voting ensemble, consisting of all taggers except Timbl (adding Timbl is slightly worse).

2 Accuracy of the original taggers

No optimization of the taggers performance was done, since the goal was not to see which tagger was most accurate but to try and improve tagging beyond the most accurate single tagger. This may give some taggers a lower score than they could achieve if optimized. Training time, tagging time and tagging accuracy measurements can be found in table 1.

3 Methods for improved accuracy

3.1 Simple voting

One straightforward way of improving accuracy is to use voting. If the errors made by the taggers were independent, voting would be very useful. For instance, three taggers, each with 95% accuracy would have an accuracy above 99% when voting. More taggers or more accurate taggers would perform even better.

Unfortunately, the errors made by the taggers are not independent. Simple voting, giving one vote to each tagger and letting a preselected tagger break ties gives 96.6% accuracy for the best ensembles. An accuracy increase from 96.0% (the best single tagger) to 96.6% is an error reduction of 15%. The difference in accuracy between the best voting ensemble and the best single tagger is significant at the 5% level (using McNemar's test (Everitt, 1977)).

Giving the taggers different voting weight manually, by for instance giving them weight proportional to their stand alone accuracy (on data separate from the test data) did not improve on simple voting.

An interesting property of voting ensembles is that for words which all taggers assign the same tag the accuracy is high. How high varies depending on how many taggers are included in the ensemble, more taggers gives fewer words were all agree, but higher accuracy on these words, see table 3.

No. of taggers	% of tokens all agree	Acc. when all agree	Acc. all words
7	87.6	99.0	96.5
6	88.4	98.9	96.5
5	89.2	98.7	96.4
4	90.2	98.5	96.2
3	91.4	98.2	96.1
2	95.3	97.6	-

Table 3: Accuracy on words for which all taggers in an ensemble assign the same tag. Accuracy for an ensemble size is measured as the mean value of the accuracy for all ensembles of that size that can be created from the examined taggers (if a tagger is not allowed to occur twice in the same ensemble).

Other properties of voting ensembles in these experiments include: Almost all voting ensembles are more accurate than the best tagger in the ensemble. The only exception is when using one of the very accurate taggers (fnTBL, Granska or TnT) together with just the two worst taggers (Stomp and Timbl), which is slightly worse than the good tagger alone.

A voting ensemble normally benefits from adding more taggers, even if the new taggers are not very good, but not always. Combining several good taggers is generally better than combining bad taggers.

A voting ensemble can suffer from adding a new very good tagger, if it is too similar to another tagger already in the ensemble (thus pretty much giving that tagger more weight). As an example of this, using just Mxpost, Timbl and Granska (or TnT) gives higher accuracy than using Mxpost, Timbl, Granska and TnT.

This is because both Granska and TnT are HMM-taggers and produce very similar results. TnT and Granska are in agreement in 97.8% of all cases, of which about 96.8% are correct. On average two taggers agree on 95.3% of all tags, with 97.6% of these being correct. In table 4 the agreement between the taggers are summarized.

This also works the other way around, adding a quite bad tagger increases the performance of an ensemble if the tagger is different enough from the taggers already in the ensemble. An example of this is

	fnTBL	Granska	Mxpost	Stomp	Timbl	TnT	TreeTagger
fnTBL	-	95.8 (97.6)	95.5 (97.9)	94.3 (97.7)	95.6 (97.3)	96.4 (97.6)	95.4 (97.7)
Granska	95.8 (97.6)	-	95.4 (97.8)	93.7 (97.8)	94.9 (97.6)	97.8 (96.8)	96.7 (96.9)
Mxpost	95.5 (97.9)	95.4 (97.8)	-	93.2 (98.2)	94.6 (97.8)	95.9 (97.8)	95.1 (97.8)
Stomp	94.3 (97.7)	93.7 (97.8)	93.2 (98.2)	-	94.9 (96.9)	94.2 (97.8)	93.7 (97.8)
Timbl	95.6 (97.3)	94.9 (97.6)	94.6 (97.8)	94.9 (96.9)	-	95.6 (97.5)	93.7 (97.8)
TnT	96.4 (97.6)	97.8 (96.8)	95.9 (97.8)	94.2 (97.8)	95.6 (97.5)	-	97.4 (96.9)
TreeTagger	95.4 (97.7)	96.7 (96.9)	95.1 (97.8)	93.7 (97.8)	94.6 (97.6)	97.4 (96.9)	-

Table 4: Agreement ratios between taggers. The percentage of words for which the taggers assign the same tag, and in parenthesis the accuracy on these words.

Stomp, which is by far the least accurate tagger, but which is very useful when added to an ensemble. For instance, an ensemble with Granska, Mxpost, TnT and Stomp has higher accuracy than an ensemble where Stomp has been exchanged for fnTBL, despite fnTBL being a much more accurate tagger.

3.2 Giving confident taggers more weight

TreeTagger, Granska and Stomp can output confidence measurements for their tag assignments, i.e. an estimate of how likely it is that an assigned tag is correct. These estimates can be used for thresholding, discarding all words where the confidence is below the threshold. In figure 1 the accuracy on the remaining words as a function of how many words are discarded is shown for the three taggers. TreeTagger gives the best estimates, while the accuracy of Granska actually decreases if a high threshold is set. This is caused by words that are unambiguous in the training data (i.e. always have the same tag), which are not necessarily actual unambiguous words. Granska is very confident on these words, while the accuracy for these words are actually below 98% in the test data.

One intuitively attractive use for these confidence measurements is to let the tagger change its voting contribution according to its confidence, i.e. give the tagger more weight for words where it is confident. This was tried in several ways. First by letting a tagger overrule the voting when confidence is above a chosen threshold, otherwise voting as normal. Second, by ignoring the vote from a tagger when the confidence is below a chosen threshold. Finally, by giving the tagger a vote proportional to the confi-

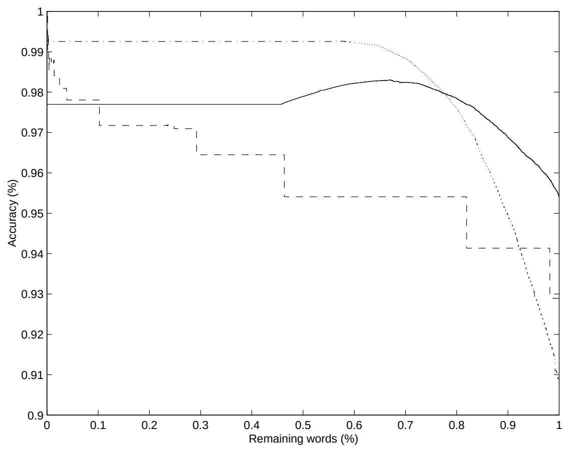

Figure 1: Plot of the accuracy on remaining tokens as a function of how many words remain when a threshold is chosen (words with scores below the threshold are discarded). The dotted line is for Stomp, the fully drawn is for Granska and the dashed and dotted line is for TreeTagger. Granska is very confident on unambiguous (in the training data) words and thus suffers when a high threshold is set, since accuracy for these are only about 98%. TreeTagger has lower accuracy here than in other experiments. For some reason TreeTagger did not choose the same tags as usual when outputting confidence estimates.

dence. This was tried for all three of the taggers above, both one at a time and all at once.

Unfortunately, this does not improve on simple voting (though the accuracy is not significantly worse either), despite the fact that for instance TreeTagger has an accuracy above 99% for some thresholds (or below 40% when ignoring low confidence words). This is caused by the fact that in general the different taggers find the same words hard (or easy) to tag. For words where TreeTagger achieve over 99% accuracy most of the other taggers also achieve about 99% accuracy. This means that they can correct each other in the few cases one of them is wrong, so voting is still superior.

The only exception to this is Stomp, which measures its confidence based on how many words of matching context it found in the training data, which when using a high enough threshold actually improves voting slightly. This is probably because it does not use the same type of information that most other taggers do (i.e. n-grams of tags), and thus does not always find the same words hard (or easy) as the other taggers. Unfortunately matches where Stomp is confident are very rare, so the increase in accuracy is very small.

Of course, the opposite is also true, words most of the taggers find easy are not always tagged very well by Stomp. This is less useful since the other taggers generally agree on the correct tag already.

3.3 Stacked classifier

Another way to combine the taggers in an ensemble is to train a new classifier on the tags selected by the taggers. This has the potential to correct even those cases where none of the taggers choose the correct tag (which no voting scheme could do). For 1.2% of all words in these experiments, no tagger is correct. It is also easy to combine taggers that use different tagsets in this way, while voting is much trickier if not all taggers use the same tagset.

For the evaluation of the stacked classifier approach, some extra work was done. First, all the taggers were trained on a training set and each then tagged an intermediary set. This intermediary set did not overlap the training set. This was then repeated for other choices of intermediary and training set, while keeping all intermediary sets disjoint. The intermediary sets were then combined into one dataset, consisting of 580 000 words, and the stacked classifiers were evaluated by 10-fold validation on this dataset.

Tagger	Accuracy (all words)
Best tagger	96.0
Timbl	96.7
Relief-F	96.8

Table 5: Accuracy of two new second level classifiers and the best single tagger. The new classifiers were trained on the output of the POS-taggers (and, in the case of Timbl, the tag of the next word). Relief-F was trained and tested on very few examples.

Training Timbl, as a memory based learner, on the output of several taggers (Granska, Mxpost, Stomp and TnT) gives 96.6% accuracy. Adding other types of information, such as the estimated probabilities of Granska and TreeTagger, decreases accuracy. This is because the method used by Timbl cannot detect that two features are dependent (i.e. use the tag TreeTagger suggests only if TreeTagger is confident). Using context increases accuracy slightly, training Timbl on the tags from fnTBL, Granska, Mxpost, Stomp, TnT and the tag selected by voting on the following word gives 96.7% accuracy.

Another stacked classifier was created using the Relief-F algorithm (Kononenko, 1994) to estimate which attributes to use when building a decision tree. Relief-F is capable of finding codependent features. It was given the following input: the tags chosen by Granska, Mxpost, TreeTagger, Stomp, TnT, Timbl (as tagger), and the confidence estimates of Granska, TreeTagger and Stomp. It achieves high accuracy, 96.8%, but this was tested on a small data set (9 000 words training data, 1 000 words test data, 10-fold validation), since Relief-F is very time consuming (weeks or months) with larger data sets.

Another way of using a stacked classifier is to specialize it on words which we believe are hard. One way to do this is to consider tags for which all taggers agree as “good enough” and the other words as hard. This will hopefully allow the stacked classifier to learn what to do for the words were voting is less successful. This reduces the problem the stacked classifier has to learn, but it also reduces the amount of available training data.

This was tried by training and testing Timbl only on those tags where there were at least two suggestions from the tagger ensemble (using 10-fold validation). Accuracy on these words was then around 78%, which gives a total accuracy of 96.6% (these word make up about 11% of all words and the accuracy on the rest is 99%). This is the same accuracy as when using Timbl as a stacked classifier on all words, so there was no improvement over a regular stacked classifier.

4 What kind of errors remain

Generally, errors occur in a "mirror" patter, i.e. if words with tag X are often misstagged as Y, then errors of misstagging type Y words as X will also be common. The most common type of tagging errors made by the taggers are choosing singular instead of plural and vice versa for nouns (7% of all errors), adjective vs. adverb (10%), determiner vs. pronoun (7%), proper noun vs. noun (7%), particle vs. adverb (5%), preposition vs. particle (4%). This corresponds well to other examinations of Swedish POS-tagging (Megyesi, 2001).

After voting there are still many errors of these types. The number of errors of some of these types actually increase. One example is the singular/plural problem for nouns. Before voting 7% of the errors belonged to this type. The number of such errors after voting would correspond to 8% of the original errors, and make up 10% of all errors that remain after voting. Before voting there are about 1 200 different error types, after voting there are only 900 types. This means that mostly uncommon errors are corrected, and that the errors that remain are concentrated to fewer categories. This is an interesting property, since it is less work to write manual correction rules for the (few) common error types than for the (many) uncommon error types.

The stacked classifiers behave similarly to voting, they mainly correct uncommon errors.

A small test shows that it is possible to write rules that correct some of the tagging errors. These rules can for instance make use of longer scope or semantic clues that the taggers cannot use. Four rules were created (by a non-linguist) in a few hours and applied on the tags selected by voting. These rules corrected 131 errors and introduced 32 new errors (and one error was changed to another error) for a net gain of 99 correct tags.

While it is possible to write rules to correct some tagging errors it seems hard to get large improvements, though using a trained linguist or more time to construct rules might achieve better results. One problem is that the common errors are mistaking one common tag for another common tag, so rules trying to correct this often introduce many new errors because there are so many correctly assigned tags of these types.

Many of the remaining errors are actually errors in the corpus, ambiguities where the suggested tag could also be considered correct or words where the correct tag can only be selected by semantic knowledge.

A discussion of common error types for automatic and manual tagging in SUC can be found in (Källgren, 1996).

5 Conclusions

Combining several taggers improves tagging accuracy, even when quite simple methods are used. An error reduction by 15% to 18% was achieved in these experiments. Using several taggers increases the computational load though, the combined classifier is at least as slow as the slowest tagger.

When combining taggers at least one good tagger should be used, then as many and as different taggers as possible should be added. Adding more taggers to an ensemble is generally good, but can sometimes decrease accuracy, if the new tagger is too similar to a tagger already in the ensemble. Dissimilar taggers are good in ensembles, even if they are not that good alone.

Some taggers can give estimates of how confident they are. Though this did not help while voting, since most taggers found the same words easy or hard, the estimates could be used for other things. One example is detecting possible errors in an existing tagged corpus. When the tag in the corpus differs from the suggested tag and the confidence of the tagger is high enough there is probably an error in the corpus. They could also be used in creating a new tagged corpus quickly, as manually checking only (the few) tags with low confidence would still catch most tagging errors. Using this kind of information for choosing when to trust the tagger has been examined before, see (Elworthy, 1994).

Words for which many taggers vote the same also have these properties, and this could be used in the same way. This also has been tried before, to remove synthetic noise (deliberately added tagging errors) in a corpus (Berthelsen and Megyesi, 2000).

Combining taggers by voting or training a new stacked classifier increases the number of errors of some of the common error types, but removes many more errors of uncommon error types. This leads to fewer total errors and a concentration of errors to fewer error types. This property is useful in several ways, it is for instance less work to manually create correction rules for a few classes of errors than for many.

6 Acknowledgments

This work has been funded by The Swedish Agency for Innovation Systems (VINNOVA).

I would like to thank my supervisor professor Viggo Kann. I would also like to thank Stockholm University for letting me use the Stockholm-Umeå Corpus.

References

Kamal M. Ali and Michael J. Pazzani. 1996. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202.

Harald Berthelsen and Beáta Megyesi. 2000. Ensemble of classifiers for noise detection in pos tagged corpora. In *Proceedings of the Third International Workshop on TEXT, SPEECH and DIALOGUE*, Brno, Czech Republic.

J. Bigert, L. Ericson, and A. Solis. 2003. Missplel and AutoEval: Two generic tools for automatic evaluation. In *Proceedings of Nodalida 2003*, Reykjavik, Iceland.

Lars Borin. 2000. Something borrowed, something blue: Rule-based combination of POS taggers. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, Athens.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference, ANLP-2000*, Seattle, USA.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, Montreal, Canada.

Eric Brill. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, Trento, IT.

Johan Carlberger and Viggo Kann. 1999. Implementing an efficient part-of-speech tagger. *Software – Practice and Experience*, 29(9):815–832.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2001. Timbl: Tilburg memory-based learner – version 4.0 reference guide.

Thomas Dietterich. 1997. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt, and Magnus Åström. 1992. The linguistic annotation system of the Stockholm-Umeå Corpus project. Technical report, Department of General Linguistics, University of Umeå (DGL-UUM-R-33), Umeå, Sweden.

David Elworthy. 1994. Automatic error detection in part of speech tagging. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester.

Brian Everitt. 1977. *The Analysis of Contingency Tables*. Chapman and Hall.

Gunnel Källgren. 1996. Linguistic indeterminacy as a source of errors in tagging. In *Proceedings of COLING-96*, Copenhagen, Denmark.

Fred Karlsson, Atro Voutilainen, Juha Heikkila, and Atro Anttila. 1995. *Constraint Grammar, A Language-independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Igor Kononenko. 1994. Estimating attributes: Analysis and extensions of RELIEF. In *Proceedings of the European Conference on Machine Learning*, Catania, Italy.

Lluís Màrquez, Lluís Padró, and Horacio Rodríguez. 1998. Improving tagging accuracy by using voting taggers. In *Proceedings of the Second Conference on Natural Language Processing and Industrial Applications, NLP+IA/TAL+AI'98*, Moncton, New Brunswick, Canada.

Lluís Màrquez, Horacio Rodríguez, Josep Carmona, and Josep Montolio. 1999. Improving POS tagging using machine–learning techniques. In *Proceedings of EMNLP/VLC'99*, Maryland, USA.

Beáta Megyesi. 2001. Comparing data-driven learning algorithms for POS tagging of Swedish. In *Proceedings of EMNLP 2001*, Carnegie Mellon University, Pittsburgh, USA.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of NAACL-2001*, Carnegie Mellon University, Pittsburgh, USA.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania, Philadelphia, USA.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Jonas Sjöbergh. 2003. Stomp, a POS-tagger with a different view. In *Proceedings of RANLP-2003*, Borovets, Bulgaria.

Kagan Tumer and Joydeep Ghosh. 1996. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–403.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 1998. Improving data driven wordclass tagging by system combination. In *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, Montreal, Canada.