Determining Ambiguity Classes for Part-of-Speech Tagging

Markus Dickinson
Indiana University
1021 E. Third Street
Bloomington, IN 47405
md7@indiana.edu

Abstract

We examine how words group together in the lexicon, in terms of ambiguity classes, and use this information in a redefined tagset to improve POS tagging. In light of errors in the training data and a limited amount of annotated data, we investigate ways to define ambiguity classes for words which consider the lexicon as a whole and predict unknown uses of words. Fitting words to typical ambiguity classes is shown to provide more accurate ambiguity classes for words and to significantly improve tagging performance.

Keywords

Part-of-speech tagging, Corpus annotation

1 Introduction

From one perspective, part of speech (POS) tagging is a task which attempts to assign a morphosyntactic label to each token in a text. From another, it is an attempt to say which word instances belong to the same class, i.e., function in the same way. The effect of this is that a tag serves to group word types together; thus, a tag can be thought of as shorthand for a set of words [5, 23]. Depending on the tagset, these word sets can be disparate; a set may contain words which are all adjectives, but some are only predicative, while some take obligatory complements. Viewed in this way, we can ask whether the POS tags in a tagset actually capture the relevant distinctions.

If the same POS tag for one collection of words behaves differently than for another, the tagset can be redefined to improve tagging (cf. [15]), given that the success of a tagger depends in part on what distinctions it learns [11, 14]. Because tags represent sets of words, to redefine a tagset, one can examine the regularities in the lexicon, in order to see whether the collections of words are appropriately grouped.

The regularities we focus on involve which ambiguity class to assign to a word, i.e., the set of "possible" tags. Ambiguity classes capture the distinctions which make tagging non-trivial. Pinpointing the most prominent classes to be disambiguated groups words with the same difficulties together and places the focus on the approximately 3% of tagging cases which a tagger gets wrong and which affect parsing (cf. [17, 10, 6]).

To determine a word's ambiguity class, it seems like we can simply extract it from the data, but this is problematic. First, because of errors in the annotated training data, a word might have too many "possible" tags, some of which are impossible. Secondly, with limited annotated data, many possible tags are never observed. Finally, even if we had sufficient error-free data, some tags are still quite rare and not completely indicative of a word. A word can mostly pattern like other words, but with some exceptions. Thus, determining what class a word belongs to becomes an issue of primary importance and the focus of this paper. This is a relevant issue not only for complete disambiguation, but also for multi-tagging tasks, where a word may have more than one tag (e.g., [6]).

We here investigate grouping words by ambiguity classes in the context of POS tagging, and we use a notion of *typicality* to overcome the three problems outlined above. Typical ambiguity classes model the regularities, ignoring the exceptions; new tags are predicted based on a word's similarity to a typical class; and tags which are atypical may be erroneous. Our starting point is a method of tagset modification for POS annotation error correction, described in section 2, since it uses ambiguity classes to deal with difficult tagging cases. In section 3, we turn to our POS tagging model: after filtering tags in section 3.1, we describe how to identify typical ambiguity classes in section 3.2 and subsequently merge classes in section 3.3, thereby predicting unknown uses of tags. For each step, we witness a gradual improvement in tagging accuracy, resulting in significant improvement. This is achieved despite basing the changes on information in the lexicon and not on contextual information.

2 Tagset modification

Using a modified tagset to deal with common ambiguities, Dickinson [12] develops a tagging method to correct POS annotation errors. Influenced by the "confusing parts of speech," or "difficult tagging distinctions," in POS annotation guidelines [21], the method is based on the idea that knowing the problematic distinction for a given corpus position can assist in tagging it.

The crucial insight is that the guideline diagnostics used, in the case of the Penn Treebank [16], to tell, e.g., RP (particle) from IN (preposition) are not the same as the ones used to tell RP from RB (adverb). These RP uses have differences in distribution based

on which distinction is involved, and thus the set of RP words can be subgrouped.

To do this, the tagset is altered while training, replacing each relevant tag with a *complex ambiguity tag*, indicating that word's *ambiguity class* and the tag at that corpus position. At a given corpus position, a word is given a complex ambiguity tag if it applies; otherwise, it retains its simple tag. This tag-splitting method (cf. [1]) results in examples like (1a) becoming (1b) in the Wall Street Journal (WSJ) part of the Penn Treebank 3.

(1) a. ago/RB
    b. ago/<IN/RB,RB>

Two constraints are used to determine the distinctions, or ambiguity classes, for words. First, low-frequency tags are filtered from consideration for an ambiguity class, in order to deal with some errors. For example, *an* uses the simple tag DT (determiner) instead tags with the class COMMA/DT because the comma tag only occurs once out of 4211 times. Secondly, only the ambiguity classes for the positions flagged by an error detection phase [13] are considered. Thus, a variation between JJ (adjective) and PRP (personal pronoun) for *ours* is not put into the model because such a variation never occurs for errors.

3 Selecting ambiguity classes

We choose to adapt this framework for POS tagging work since the emphasis on ambiguity classes finds regularities beyond the distinctions in the tagset. POS tagging, however, differs in crucial ways from error correction. First, training data and testing data are disjoint for tagging, whereas they are identical for error correction (i.e., the entire corpus is used for both), forcing us to consider unknown uses of words in ambiguity class assignment. Secondly, whereas automatic correction focuses only on positions flagged as potential errors, POS tagging is for an entire text, giving a large number of distinctions. In assigning ambiguity classes for POS tagging, therefore, we need new criteria to determine what words group together. Instead of asking whether it is involved in an error, we suggest typicality as a criterion for the relevance of an ambiguity class: is it a common distinction?

Following Toutanova et al. [25], we use the WSJ corpus merged data, sections 00-18 for training, sections 19-21 for development, and sections 22-24 for testing. All tagset modification is done to training data only, and tags are mapped back to Penn Treebank tags for evaluating tagger precision (see section 4.2).

We could assign ambiguity classes based on all possible tags for a word (cf. [7]), but this will not generalize well. The problem is that this method results in too many specific classes, which serves to isolate words in the lexicon. With 280 ambiguity classes and 887 total tags, we find unique classes like JJ/JJR/RB/RBR/VB for the word *further*. To better group words together, we need to limit what ambiguity classes are possible.

In fact, we observe that adding data makes this problem worse. We cumulatively calculated the set of ambiguity classes in the corpus, section by section, as shown in figure 1. The set of ambiguity classes grows indefinitely, albeit slowly. As more and more instances are added to the corpus, there is a greater tendency for rare cases to emerge and for errors to be introduced.

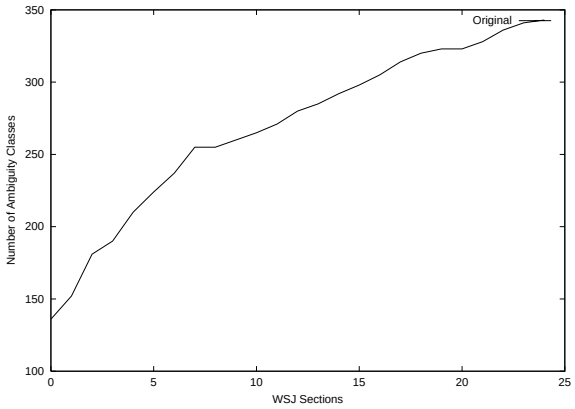

Fig. 1: *Growth rate of ambiguity classes*

Our task thus becomes one of restricting the ambiguity class for each word. We find that restrictions specific to an individual word (e.g., filtering) are insufficient (section 3.1), requiring global restrictions which consider the lexicon as a whole (section 3.2). Fitting words to these global, typical classes indicates which unseen tags are appropriate (section 3.3).

3.1 Filtering

The first restriction directly addresses the problems of erroneous tags and low-frequency tags that are not indicative of a word by filtering out tags occurring less than 10% of the time for a word (cf. [9]). As an example of how this handles errors, instead of CD/DT/JJ/NN/NNP/VBP for *the*—which has five erroneous tags—we have only the correct DT. As an example of a non-indicative tag, the word *all* varies between DT, PDT (predeterminer), and RB, but RB accounts for 3.7% of the cases (38/1017); after filtering, we obtain DT/PDT. It is not that RB is wrong; it is that DT and PDT are its most prototypical uses and by restricting our focus to only DT and PDT, we are able to group *all* with its capitalized form *All*. Thus, *all* now has three possible tags: <DT/PDT,DT>, <DT/PDT,PDT>, and RB.

This tagging model uses 155 ambiguity classes, but two problems remain. First, we still have some highly specific classes; many words with similarities to other words pattern only like themselves. For instance, *Put* (which only appears 17 times) is alone in having the ambiguity class JJ/NN/VB/VBD/VBN. Furthermore, many classes seem ad hoc; ambiguity classes like \$/NNP (for the token *C*) are not problematic variations for annotators [21]. Secondly, it is not clear how filtering by itself is a sufficient test of indicativeness.

3.2 Identifying typical tag classes

Filtering is a local task, but to evaluate whether a word's ambiguity class captures a true regularity, i.e., is like others, we need to consider the whole lexicon. Since we want to capture regularities—i.e., repeated uses of the same class—we use a frequency-based criterion, to determine if an ambiguity class is *typical*.

For this (*Typical*) model, after filtering tags, we keep only the ambiguity classes with more than n tokens, where n is empirically determined. So, a class such as JJ/RB, which has 59 word types realized with 5054 tokens, is ranked above NN/NNP (common/proper noun), with 378 types and 4217 tokens. Such a token-based measure best reflects the prevalent patterns in the training corpus (i.e., the typical classes) and which decisions the tagger sees repeatedly. For the remainder of this discussion, we will use the model with n equal to 400, which uses 38 ambiguity classes (see section 4 for a comparison of different values of n).

As a side effect, we discover that some regularities are word-specific; in other words, some classes are essentially lexicalized. For example, *that* is the only DT/IN/WDT word, with 7699 tokens, and thus tagging an item as <DT/IN/WDT,DT> is the same as tagging it <*that*,DT>. Others have shown such lexicalization to be useful in tagging (e.g., [20]). Our approach differs by automatically finding words which do not pattern with any others, and because we filter out non-indicative tags, there is a slight difference between our "lexicalized" classes and the general notion of lexicalization. Consider the word *like*, the only IN/VB word; in our model, it has four possible tags: <IN/VB,IN>, <IN/VB,VB>, JJ, and VBP. The JJ and VBP (present tense verb) cases get grouped with other JJ and VBP cases, instead of receiving the tags <*like*,JJ> and <*like*,VBP>. It patterns uniquely in being the only word which can be a preposition (IN) and a verb (VB); the other uses can be grouped with other corpus instances.

Examining the lexicon This model is an improvement, but it is inadequate. Similar words are often in different classes. For example, *explained* is VBD/VBN (past tense verb/past participle),while *classified* is JJ/VBD/VBN and *accomplished* is JJ/VBN, yet all seem to have the same possibilities. JJ never appears in the training data for *explained*, yet it is a possible tag. Consider also a word like *accepted*: it varies between JJ, VBD, and VBN, yet JJ is only 1 of the 41 occurrences, so the ambiguity class becomes VBD/VBN after filtering. Yet, this instance of JJ is correct (*futures have become an accepted/JJ part of the financial landscape*). In cases like this, it becomes apparent that basing indicativeness on individual frequency is inadequate: the tag is neither more nor less indicative of *accepted* than VBD or VBN. Many verbs that are VBD/VBN, whether because of filtering (cf. *accepted*) or lack of observation (cf. *explained*), should also have JJ as a possible tag.

We thus want some way to predict tags not observed in the training data and to overcome excessive filtering. The solution seems to be in performing a limited amount of merging of classes; for example, JJ/VBN, VBD/VBN, and JJ/VBD/VBN can be combined into the superset class JJ/VBD/VBN.

3.3 Merging ambiguity classes

After grouping words by typical ambiguity classes, we then merge classes, based on which tags are predictable from which other tags (*Merge* model). To find the mappings from one ambiguity class into another, superset class, we calculated the ambiguity class for every word in a portion of the training data (sections 00-15), and observed which tags are added for each word in some held-out data (sections 16-18). For example, 18 NN/VB word types become NN/VB/VBP words when adding more data. With 16 sections for the base set of ambiguity classes, this ensures a relatively stable set of ambiguity classes, and using the held-out data ensures that we capture the relevant property: which tag is predictable from an ambiguity class? Once we automatically deduced the mappings (e.g., NN/VB $\mapsto$ NN/VB/VBP), we use them to merge ambiguity classes together.

To account for noise and idiosyncratic behavior, we use a few simple restrictions: 1) The resulting ambiguity class must be a typical class. 2) The mapping occurs for at least two words in the held-out data since single-occurring mappings are not general. 3) The class is not very fertile, i.e., does not generate lots of tag possibilities. Specifically, no more than three other tags are allowed. 4) Only the highest-ranking mapping is used. For example, the twice-occurring VB/VBP $\mapsto$ IN/VB/VBP is not used because VB/VBP already has a mapping. With this method, we also merge single tags into ambiguity classes—e.g., VB $\mapsto$ VB/VBP. The full set of mappings can be seen in figure 2.

Original	New
NNPS	NNPS/NNS
VB	VB/VBP
VBP	VB/VBP
VBD	VBD/VBN
VBN	VBD/VBN
VBG	NN/VBG
VBZ	NNS/VBZ
JJ/VBN	JJ/VBD/VBN
VBD/VBN	JJ/VBD/VBN
NN/VB	NN/VB/VBP
NN/VBP	NN/VB/VBP
VB/VBP	NN/VB/VBP
NNP/NNPS	NNP/NNPS/NNS
NNP/NNS	NNP/NNPS/NNS

Fig. 2: *Mappings for merging classes*

This merging serves to counteract some filtering, by putting some filtered tags back into ambiguity classes. On the one hand, we filtered JJ from the ambiguity class for *accepted*, making it VBD/VBN, because JJ appears only once out of 41 times. Now, it gets put back in, making the class JJ/VBD/VBN. On the other hand, we filtered RBR (comparative adverb) from the ambiguity class of *trimmed*, making it VBD/VBN, because it occurs once out of 15 times: RBR is erroneous, and it stays out of the ambiguity class. By using a criterion other than frequency, we can begin to separate errors from rare instances, giving a good first step in

having more selective filtering, instead of simply filtering out low-frequency tags from a tagging model (e.g., [9, 22, 4]). This is especially important for methods which depend upon rare but correct instances [8].

Additionally, we predict tags for words which never had that tag in the data, but should have. The example of *trimmed* is another case where JJ is appropriate, and its class becomes JJ/VBD/VBN.

This method of merging can obviously be improved. We almost definitely will over-generalize since we do not take morphology or tag distributions into account—e.g., not all VBD words are also VBN (cf. *went*). Still, it is important to remember that these assignments are currently being used only as an indication of possibilities; in most contexts, we do not expect VBN to be a legitimate tag for *went*.

Tag prediction Merged ambiguity classes can now predict the presence of possible tags for a word because they may contain tags a word lacks. To add these tags directly to a tagging model is straightforward for a tagger with a transparent lexicon. For every word with a complex ambiguity class, we add a count of one for any tag which is predicted to appear but does not (*Merge+*). For example, the word *cheer* originally varied between NN and VB, with one occurrence of each. We now add a count of one for VBP since NN/VB/VBP is its merged ambiguity class. A count of one makes the tagger aware that this tag is possible without making any further claims.

The prediction of unknown tag uses is in the spirit of Toutanova and Manning [26], who "augmented [a tag dictionary] so as to capture a few basic systematic tag regularities that are found in English. Namely, for regular verbs the *-ed* form can be either a VBD or a VBN and similarly the stem form can be either a VBP or VB." Our predictions, however, arise from a data-driven analysis of word groupings, instead of being hand-encoded. Either way, lexicon augmentation can be used as a sanity check on filtering noisy data.

4 Evaluation

There are two ways to evaluate the resulting models. First, to gauge whether the ambiguity classes are capturing true facts about these words, or whether they are over- or under-generalizing, some degree of qualitative analysis is needed. Secondly, to gauge the effectiveness of better groupings in the lexicon, we will see how the ambiguity classes affect the quality of POS tagging. This is only one way to use these groupings, however; given the confounding factor of being integrated into an already complicated tagging model, both kinds of evaluation are important.

4.1 Quality of ambiguity classes

To determine the quality of the ambiguity classes used, we need a test bed of words with all of their truly possible tags. Thus, we sampled 100 lexical entries (from sections 00-18), removed their tags, and hand-annotated the set of possible tags. To guide this process, we first gathered the list of all (unaltered) ambiguity classes from the lexicon, so that the annotator could first mark a word's most prominent tags and then consult the list to see which other tags are general possibilities.

We then took the entries from the original lexicon for the 100 words and compared their possible tags to the hand-created set. We found that 49 words matched this set, while 51 were missing tags. Thus, we can see that the task of predicting tags for known words is a high priority for POS lexicon coverage: over half the word types are missing at least one tag.

The Merge(+) model, on the other hand, has 39 such undergeneralizations with only one overgeneralization (*describes*, predicted to be NNS/VBZ instead of VBZ only), correctly changing words like *smile* from NN/VB to NN/VB/VBP and *bottling* from VBG to NN/VBG. There were also six cases which were closer to a correct distribution, even if they were still missing tags. The word *responding*, for example, was originally VBG, but is now NN/VBG; although its complete set of possible tags is JJ/NN/VBG, it is now improved. With 18 total improved words, we are successfully adding more possible tags, without adding much noise.

4.2 POS tagging results

Having shown that the ambiguity classes are successfully capturing the range of a word's tag possibilities, we want to test the effectiveness of using them to group words for POS tagging. Corpus positions are assigned complex ambiguity tags where appropriate for training, using the splitting framework from section 2, and the complex tags are mapped back to their simple tags for evaluation. Thus, if the tagger assigns *ago* the tag <IN/RB,RB>, we map it to RB in order to compare it against the benchmark.

Development data As a baseline for the development data, the default version of the Hidden Markov Model (HMM) tagger TnT [3] obtains a precision of 96.48%. Using filtering at 10% to assign ambiguity classes for tag splitting, the tagging model has 96.63% precision on the development data, showing that a first pass at using ambiguity classes provides better performance.

Testing the different ambiguity class models, we present the results side-by-side in figure 3. The best results are for the Merge+ model, with a token cutoff of 400, giving a precision of 96.71%, an improvement gained by making fewer, more general classes than the Typical model and by extrapolating the ambiguity classes directly to the lexical entries. It is also important to note the overall trends: each model slightly improves upon the previous one, for all cutoff levels.

Testing data After developing the different models, we ran them on the testing data (sections 22-24 of the WSJ) for $n = 400$. As shown in figure 4, we see the same improvements as with the development data, demonstrating that the improvements are not specific to one data set. Using McNemar's Test [18], the results for Merge+ are significantly higher ($p < .001$) than for the Baseline.

	Typical		Merge		Merge+
n	Pre.	AC	Pre.	AC	Pre.
100	96.64%	56	96.65%	48	96.68%
200	96.64%	47	96.65%	41	96.69%
300	96.66%	42	96.66%	36	96.70%
400	96.66%	38	96.67%	33	**96.71%**
500	96.65%	34	96.66%	30	96.70%

Fig. 3: *Results on sections 19-21 of the WSJ (Pre. = precision, AC = number of ambiguity classes)*

Model	Development	Testing
Baseline	96.48%	96.46%
Typical	96.66%	96.65%
Merge	96.67%	96.66%
Merge+	96.71%	96.70%

Fig. 4: *Results on sections 22-24 of the WSJ*

We also wanted to see how applicable our models are to other genres of text. As with better unknown word tagging, predicting unknown uses of known words mitigates the need for more training data, by filling in some gaps of what has not been observed. Such methods are potentially more applicable to other genres of text: tag uses in one genre may not appear in another. Thus, we tested the models on the Brown corpus part of the Penn Treebank, as shown in figure 5. Using McNemar's Test, the results for Merge+ are significantly higher ($p < .001$) than for the Baseline.

Model	Precision
Baseline	94.60%
Typical	94.79%
Merge	94.81%
Merge+	94.93%

Fig. 5: *Results on the Brown corpus*

We see the same trends here, showing the methods developed here improve even on another corpus. The percentage gain is still somewhat small—0.33% from Baseline to Merge+—but with a larger corpus, we can better see the impact of the improvement, obtaining a reduction of 1550 errors (24,815-23,265).

Discussion The increase in tagging precision, from 96.46% to 96.70% on the testing data, is only 0.24% and is below the state-of-the-art precision of 97.33% by Shen et al. [24] on the same data. What is important to notice, however, is not the absolute accuracy of the particular method used here, but that we have seen significant improvement by examining how words group in the lexicon. Further, we have improved coverage of the lexicon by fitting words to typical ambiguity classes. This has the potential to make any tagger better represent possible tags for words.

We have obtained an improvement in performance without encoding a variety of features or changing a tagging algorithm. In fact, we have only generalized patterns of tags found across the lexicon; we have not used any contextual information. The principles behind these techniques are applicable to any tagging method; how they are applied to a tagger depends upon the tagger, however. The tagset alteration method works with HMMs because there is a direct interpretation of tag splitting and merging, namely that they correspond to state splitting and merging. For a decision tree tagger, it might be best to use the ambiguity classes as nodes in the decision tree (cf. [17]). Similarly, it is not yet clear how these techniques interact with tagging methods which have their own smoothing and error correction capabilities.

A criticism of this work might be that it is language-specific or tagset-specific. In that all languages have ambiguous words, the claim about language-specificity has to be empirically determined. However, taggers which encode highly specific features are language-specific (and likely tagset-specific). Consider, for example, how Toutanova and Manning [26] determine whether a word is a particle (RP): "the current word is often used as a particle, and ... there is a verb at most 3 positions to the left, which is 'known' to have a good chance of taking the current word as a particle." This language-specificity is not altogether a bad thing, as general tagging algorithms have, at least for English, seemed to have hit an upper bound, and language-specific features may be necessary to improve. The approach outlined here, however, uses no hand-encoded knowledge and does not increase the complexity of the tagging algorithm.

As for tagset-specificity, which ambiguity class a word has is clearly dependent upon the tagset used, but it is less clear how these methods work with tagsets having different degrees of ambiguity or capturing different morphosyntactic properties. Modifying these methods for another tagset could tell us about how its tags interact and whether a better organization of the lexicon is needed. Using a corpus with a tagset that can be mapped to smaller tagsets (see, e.g., [2]) could more precisely determine the properties which make this tagset modification successful.

5 Summary and Outlook

We have investigated ways to assign ambiguity classes to words in order to overcome errors in the training data and a limited amount of annotated training data, thereby leading to a more robust lexicon and improvements in POS tagging. In order to make ambiguity class definitions work: 1) we defined typical ambiguity classes based on their frequency of occurrence, merging classes when appropriate; and 2) we made individual words conform to these classes, by using filtering and adding missing counts to lexical entries.

A benefit of the method is that it can target words or ambiguities of interest. After all, classification by ambiguity classes works well when narrowing in on error classes. Future work can investigate exactly which classes are useful for POS tagging and why, specifically examining whether these more specific tags provide more informative contexts (cf. [2, 12]). Experimenting on other tagsets and corpora can test the effects of tagset design [11] and provide feedback on annotation schemes. This examination can also help

address the rather arbitrary thresholds used for ambiguity class selection. Instead, one can attempt to more robustly cluster words in the lexicon, using not only lexicon information, but contextual information to distinguish the ambiguity classes (cf. [23]). Additionally, given that there are annotation errors in the evaluation data, qualitative analysis of the tagging results is needed [19].

Aside from continuing to improve the assignment of ambiguity classes, this work could impact other POS taggers where an ambiguity class represents a word or is a feature (e.g., [7, 9]), or where a word is assigned more than one tag [6]. These methods could also be adapted for any annotation task using a lexicon.

Acknowledgments Thanks to Adriane Boyd and Detmar Meurers for comments on an earlier draft and to Stephanie Dickinson for statistical advising. This material is based upon work supported by the National Science Foundation under Grant No. IIS-0623837.

References

[1] T. Brants. Estimating markov model structures. In *Proceedings ICSLP-96*, pages 893–896, Philadelphia, PA, 1996.

[2] T. Brants. Internal and external tagsets in part-of-speech tagging. In *Proceedings of Eurospeech*, Rhodes, Greece, 1997.

[3] T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of ANLP 2000*, pages 224–231, Seattle, WA, 2000.

[4] E. Brill and M. Pop. Unsupervised learning of disambiguation rules for part of speech tagging. In K. W. Church, editor, *Natural Language Processing Using Very Large Corpora*, pages 27–42. Kluwer Academic Press, Dordrecht, 1999.

[5] A. Clark. Combining distributional and morphological information for part of speech induction. In *Proceedings of EACL-03*, pages 59–66, 2003.

[6] J. R. Curran, S. Clark, and D. Vadas. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of ACL-06*, pages 697–704, 2006.

[7] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of ANLP-92*, pages 133–140, Trento, Italy, 1992.

[8] W. Daelemans, A. van den Bosch, and J. Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11–41, 1999.

[9] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. MBT: A memory-based part of speech tagger-generator. In *Proceedings of VLC-96*, pages 14–27, Copenhagen, 1996.

[10] M. Dalrymple. How much can part of speech tagging help parsing? *Natural Language Engineering*, 12(4):373–389, 2006.

[11] H. Déjean. How to evaluate and compare tagsets? a proposal. In *Proceedings of LREC-00*, Athens, 2000.

[12] M. Dickinson. From detecting errors to automatically correcting them. In *Proceedings of EACL-06*, pages 265–272, Trento, Italy, 2006.

[13] M. Dickinson and W. D. Meurers. Detecting errors in part-of-speech annotation. In *Proceedings of EACL-03*, pages 107–114, Budapest, Hungary, 2003.

[14] D. Elworthy. Tagset design and inflected languages. In *Proceedings of the ACL-SIGDAT Workshop*, Dublin, 1995.

[15] A. MacKinlay and T. Baldwin. Pos tagging with a more informative tagset. In *Proceedings of ALTW 2005*, pages 40–48, Sydney, Australia, 2005.

[16] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[17] L. Marquez, L. Padro, and H. Rodriguez. A machine learning approach to POS tagging. *Machine Learning*, 39(1):59–91, 2000.

[18] Q. McNemar. Note on the sampling error of the difference between correlated proportions. *Psychometrika*, 12:153–157, 1947.

[19] L. Padro and L. Marquez. On the evaluation and comparison of taggers: the effect of noise in testing corpora. In *Proceedings of COLING/ACL-98*, pages 997–1002, 1998.

[20] F. Pla and A. Molina. Improving part-of-speech tagging using lexicalized HMMs. *Natural Language Engineering*, 10(2):167–189, 2004.

[21] B. Santorini. Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision, 2nd printing). Technical Report MS-CIS-90-47, The University of Pennsylvania, Philadelphia, PA, June 1990.

[22] H. Schmid. Part-of-speech tagging with neural networks. In *Proceedings of COLING 94*, pages 172–176, Kyoto, Japan, 1994.

[23] H. Schütze. Distributional part-of-speech tagging. In *Proceedings of EACL-95*, pages 141–148, Dublin, Ireland, 1995.

[24] L. Shen, G. Satta, and A. K. Joshi. Guided learning for bidirectional sequence classification. In *Proceedings of ACL-07*, pages 760–767, 2007.

[25] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging using a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259, 2003.

[26] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of EMNLP/VLC-2000*, Hong Kong, 2000.