In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03). Trento, Italy.

From detecting errors to automatically correcting them

Markus Dickinson
Department of Linguistics
Georgetown University
mad87@georgetown.edu

Abstract

Faced with the problem of annotation errors in part-of-speech (POS) annotated corpora, we develop a method for automatically correcting such errors. Building on top of a successful error detection method, we first try correcting a corpus using two off-the-shelf POS taggers, based on the idea that they enforce consistency; with this, we find some improvement. After some discussion of the tagging process, we alter the tagging model to better account for problematic tagging distinctions. This modification results in significantly improved performance, reducing the error rate of the corpus.

1 Introduction

Annotated corpora serve as training material and as "gold standard" testing material for the development of tools in computational linguistics, and as a source of data for theoretical linguists searching for relevant language patterns. However, they contain annotation errors, and such errors provide unreliable training and evaluation data, as has been previously shown (see ch. 1 of Dickinson (2005) and references therein). Improving the quality of linguistic annotation where possible is thus a key issue for the use of annotated corpora in computational and theoretical linguistics.

Research has gone into automatically detecting annotation errors for part-of-speech annotation (van Halteren, 2000; Květoň and Oliva, 2002; Dickinson and Meurers, 2003), yet there has been virtually no work on automatically or semi-automatically correcting such annotation errors.[1] Automatic correction can speed up corpus improvement efforts and provide new data for NLP technology training on the corpus. Additionally, an investigation into automatic correction forces us to re-evaluate the technology using the corpus, providing new insights into such technology.

We propose in this paper to automatically correct part-of-speech (POS) annotation errors in corpora, by adapting existing technology for POS disambiguation. We build the correction work on top of a POS error detection phase, described in section 2. In section 3 we discuss how to evaluate corpus correction work, given that we have no benchmark corpus to compare with. We turn to the actual work of correction in section 4, using two different POS taggers as automatic correctors and using the Wall Street Journal (WSJ) corpus as our data. After more thoroughly investigating how problematic tagging distinctions affect the POS disambiguation task, in section 5 we modify the tagging model in order to better account for these distinctions, and we show this to significantly reduce the error rate of a corpus.

It might be objected that automatic correction of annotation errors will cause information to be lost or will make the corpus worse than it was, but the construction of a large corpus generally requires semi-automated methods of annotation, and automatic tools must be used sensibly at every stage in the corpus building process. Automated annotation methods are not perfect, but humans also add errors, from biases and inconsistent judgments. Thus, automatic corpus correction methods can be used semi-automatically, just as the original corpus creation methods were used.

[1] Oliva (2001) specifies hand-written rules to detect and then correct errors, but there is no general correction scheme.

2 Detecting POS Annotation Errors

To correct part-of-speech (POS) annotation errors, one has to first detect such errors. Although there are POS error detection approaches, using, e.g., anomaly detection (Eskin, 2000), our approach builds on the variation n-gram algorithm introduced in Dickinson and Meurers (2003) and Dickinson (2005). As we will show in section 5, such a method is useful for correction because it highlights recurring problematic tag distinctions in the corpus.

The idea behind the variation n-gram approach is that a string occurring more than once can occur with different labels in a corpus, which is referred to as *variation*. Variation is caused by one of two reasons: i) *ambiguity*: there is a type of string with multiple possible labels and different corpus occurrences of that string realize the different options, or ii) *error*: the tagging of a string is inconsistent across comparable occurrences.

The more similar the context of a variation, the more likely the variation is an error. In Dickinson and Meurers (2003), contexts are composed of words, and identity of the context is required. The term *variation n-gram* refers to an n-gram (of words) in a corpus that contains a string annotated differently in another occurrence of the same n-gram in the corpus. The string exhibiting the variation is referred to as the *variation nucleus*.

For example, in the WSJ corpus, part of the Penn Treebank 3 release (Marcus et al., 1993), the string in (1) is a variation 12-gram since *off* is a variation nucleus that in one corpus occurrence is tagged as a preposition (IN), while in another it is tagged as a particle (RP).

(1) to ward off a hostile takeover attempt by two European shipping concerns

Once the variation n-grams for a corpus have been computed, heuristics are employed to classify the variations into errors and ambiguities. The most effective heuristic takes into account the fact that natural languages favor the use of local dependencies over non-local ones: nuclei found at the fringe of an n-gram are more likely to be genuine ambiguities than those occurring with at least one word of surrounding context.

Running the variation n-gram error detection method on the WSJ turns up 7141 distinct[2] non-fringe nuclei, of which an estimated 92.8%, or 6627, are erroneous.[3] Since a variation nucleus refers to multiple corpus positions, this precision is a precision on types; we, however, are correcting tokens. Still, this precision is high enough to experiment with error correction.

3 Methodology

Since we intend to correct a corpus with POS annotation errors, we have no true benchmark by which to gauge the accuracy of the corrected corpus, and we thus created a hand-checked subcorpus. Using the variation n-gram output, we flagged every non-fringe variation nucleus (token) as a potential error, giving us 21,575 flagged positions in the WSJ. From this set, we sampled 300 positions, removed the tag for each position, and hand-marked what the correct tag should be, based solely on the tagset definitions given in the WSJ tagging manual (Santorini, 1990), i.e., blind to the original data. Because some of the tagset distinctions were not defined clearly enough in the guidelines, in 20 cases we could not decide what the exact tag should be. For the purposes of comparison, we score a match with either tag as correct since a human could not disambiguate such cases.

For the benchmark, we find that 201 positions in our sample set of 300 are correct, giving us a precision of 67%. A correction method must then surpass this precision figure in order to be useful.

4 Approach to correction

Since our error detection phase relies on variation in annotation, i.e., the inconsistent application of POS labels across the corpus, we propose to correct such errors by enforcing consistency in the text. As van Halteren (2000) points out, POS taggers can be used to enforce consistency, and so we employ off-the-shelf supervised POS taggers for error correction. The procedure is as follows:

1. Train the tagger on the entire corpus.
2. Run the trained tagger over the same corpus.
3. For the positions the variation n-gram detection method flags as potentially erroneous, choose the label obtained in step 2.

We do not split training data from testing data because we want to apply the patterns found in the

[2] Being distinct means each corpus position is only taken into account for the longest variation n-gram it occurs in.

[3] The recall cannot easily be estimated, but this is still a significant number of errors.

whole corpus to the corpus we want to correct, which happens to be the same corpus.[4] If the tagger has learned the consistent patterns in the corpus, it will then generalize these patterns to the problematic parts of the corpus.

This approach hinges on high-quality error detection since in general we cannot assume that discrepancies between a POS tagger and the benchmark are errors in the benchmark. Van Halteren (2000), for example, found that his tagger was correct in only 20% of disagreements with the benchmark. By focusing only on the variation-flagged positions, we expect the tagger decisions to be more often correct than incorrect.

We use two off-the-shelf taggers for correction, the Markov model tagger TnT (Brants, 2000) and the Decision Tree Tagger (Schmid, 1997), which we will abbreviate as DTT. Both taggers use probabilistic contextual and lexical information to disambiguate a tag at a particular corpus position. The difference is that TnT obtains contextual probabilities from maximum likelihood counts, whereas DTT constructs binary-branching decision trees to obtain contextual probabilities. In both cases, instead of looking at n-grams of words, the taggers use n-grams of tags. This generalization is desirable, as the variation n-gram method shows that the corpus has conflicting labels for the exact same sequence of n words.

Results For the TnT tagger, we obtain an overall precision of 71.67% (215/300) on the 300 hand-annotated samples. For the DTT tagger, we get a higher precision, that of 76.33% (229/300). The DTT results are a significant improvement over the original corpus precision of 67% ($p = .004$[5]), while the TnT results are not.

As mentioned, tagger-benchmark disagreements are more commonly tagger errors, but we find the opposite for variation-flagged positions. Narrowing in on the positions which the tagger changed, we find a precision of 58.56% (65/111) for TnT and 65.59% (69/107) for DTT. As the goal of correction is to change tags with 100% accuracy, we place a priority in improving these figures.

One likely reason that DTT outperforms TnT is its more flexible context. For instance, in example (2)—which DTT correctly changes and TnT does not— to know that *such* should be changed from adjective (JJ) to pre-determiner (PDT), one only need look at the following determiner (DT) *an*, and that provides enough context to disambiguate. TnT uses a fixed context of trigrams, and so can be swayed by irrelevant tags—here, the previous tags—which DTT can in principle ignore.[6]

(2) Mr. Bush was n't interested in such/JJ an informal get-together .

5 Modifying the tagging model

The errors detected by the variation n-gram method arise from variation in the corpus, often reflecting decisions difficult for annotators to maintain over the entire corpus, for example, the distinction between preposition (IN) and particle (RP) (as in (1)). Although these distinctions are listed in the tagging guidelines (Santorini, 1990), nowhere are they encoded in the tags themselves; thus, a tagger has no direct way of knowing that IN and RP are easily confusable but IN and NN (common noun) are not. In order to improve automatic correction, we can add information about these recurring distinctions to the tagging model, making the tagger aware of the difficult distinctions. But how do we make a tagger "aware" of a relevant problematic distinction?

Consider the domain of POS tagging. Every word patterns uniquely, yet there are generalizations about words which we capture by grouping them into POS classes. By grouping words into the same class, there is often a claim that these words share distributional properties. But how true this is depends on one's tagset (see, e.g., Déjean (2000)). If we can alter the tagset to better match the distributional facts, we can improve correction.

To see how problematic distinctions can assist in altering the tagset, consider the words *away* and *aboard*, both of which can be adverbs (RB) in the Penn Treebank, as shown in (3a) and (4a). In example (3b), we find that *away* can also be a particle (RP), thus making it a part of the *ambiguity class* RB/RP. On the other hand, as shown in (4b), *aboard* can be a preposition (IN), but not a particle, putting it in the ambiguity class IN/RB. Crucially, not only do *away* and *aboard* belong

[4] Note, then, that some typical tagging issues, such as dealing with unknown words, are not an issue for us.

[5] All p-values in this paper are from McNemar's Test (McNemar, 1947) for analyzing matched dichotomous data (i.e., a correct or incorrect score for each corpus position from both models).

[6] As DTT does not provide a way of viewing output trees, we cannot confirm that this is the reason for improvement.

to different ambiguity classes, but their adverbial uses are also distinguished. The adverbial *away* is followed by *from*, a construction forbidden for *aboard*. When we examine the RB/RP words, we find that they form a natural class: *apart*, *aside*, and *away*, all of which can be followed by *from*.

(3) a. the Cray-3 machine is at least another year away/RB from a ... prototype
b. A lot of people think 0 I will give away/RP the store

(4) a. Saturday 's crash ... that *T* killed 132 of the 146 people aboard/RB
b. These are used * aboard/IN military helicopters

Although not every ambiguity class is so cleanly delineated, this example demonstrates that such classes can be used to redefine a tagging model with more unified groupings.

5.1 Using complex ambiguity tags

We thus propose splitting a class such as RB into subclasses, using these ambiguity classes—JJ/RB, NN/RB, IN/RB, etc.—akin to previous work on splitting labels in order to obtain better statistics (e.g., Brants (1996); Ule (2003)) for situations with "the same label but different usage" (Ule, 2003, p. 181). By taking this approach, we are narrowing in on what annotators were instructed to focus on, namely "difficult tagging decisions," (Santorini, 1990, p. 7).

We implement this idea by assigning words a new, complex tag composed of its ambiguity class and the benchmark tag for that position. For example, *ago* has the ambiguity class IN/RB, and in example (5a), it resolves to RB. Thus, following the notation in Pla and Molina (2004), we assign *ago* the *complex ambiguity tag* <IN/RB,RB> in the training data, as shown in (5b).

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

Complex ambiguity tags can provide better distinctions than the unaltered tags. For example, words which vary between IN and RB and tagged as IN (e.g., *ago*, tagged <IN/RB,IN>) can ignore the contextual information that words varying between DT (determiner) and IN (e.g., *that*, tagged <DT/IN,IN>) provide. This proposal is in the spirit of a tagger like that described in Marquez et al (2000), which breaks the POS tagging problem into one problem for each ambiguity class, but because we alter the tagset here, different underlying tagging algorithms can be used.

To take an example, consider the 5-gram *revenue of about $ 370* as it is tagged by TnT. The 5-gram (at position 1344) in the WSJ is annotated as in (6). The tag for *about* is incorrect since "*about* when used to mean 'approximately' should be tagged as an adverb (RB), rather than a preposition (IN)" (Santorini, 1990, p. 22).

(6) revenue/NN of/IN about/IN $/$ 370/CD

Between *of* and *$*, the word *about* varies between preposition (IN) and adverb (RB): it is IN 67 times and RB 65 times. After training TnT on the original corpus, we find that RB is a slightly better predictor of the following $ tag, as shown in (7), but, due to the surrounding probabilities, IN is the tag TnT assigns.

(7) a. $p(\$|\text{IN,RB}) = .0859$
b. $p(\$|\text{IN,IN}) = .0635$

The difference between probabilities is more pronounced in the model with complex ambiguity tags. The word *about* generally varies between three tags: IN, RB, and RP (particle), receiving the ambiguity class IN/RB/RP (as *of* also does). For IN/RB/RP words, RB is significantly more probable in this context than IN, as shown in (8).

(8) a. $p(\$|\text{<IN/RB/RP,IN>,<IN/RB/RP,RB>}) = .6016$
b. $p(\$|\text{<IN/RB/RP,IN>,<IN/RB/RP,IN>}) = .1256$

Comparing (7) and (8), we see that RB for the ambiguity class of IN/RB/RP behaves differently than the general class of RB words.

We have just shown that the contextual probabilities of an n-gram tagger are affected when using complex ambiguity tags; lexical probabilities are also dramatically changed. The relevant probabilities were originally as in (9), but for the modified corpus, we have the probabilities in (10).

(9) a. $p(about|\text{IN}) = 2074/134926 = .0154$
b. $p(about|\text{RB}) = 785/42207 = .0186$

(10) a. $p(about|\text{<IN/RB/RP,IN>}) = 2074/64046 = .0324$
b. $p(about|\text{<IN/RB/RP,RB>}) = 785/2045 = .3839$

These altered probabilities provide information similar to that found in a lexicalized tagger—i.e., *about* behaves differently than the rest of its class—but the altered contextual probabilities, unlike a lexicalized tagger, bring general IN/RB/RP class information to bear on this tagging situation. Combining the two, we get the correct tag RB at this position.

Since variation errors are errors for words with prominent ambiguity classes, zeroing in on these ambiguity classes should provide more accurate probabilities. For this to work, however, we have to ensure that we have the most effective ambiguity class for every word.

5.2 Assigning complex ambiguity tags

In the tagging literature (e.g., Cutting et al (1992)) an ambiguity class is often composed of the set of every possible tag for a word. For correction, using every possible tag for an ambiguity class will result in too many classes, for two reasons: 1) there are erroneous tags which should not be part of the ambiguity class, and 2) some classes are irrelevant for disambiguating variation positions.

Guided by these considerations, we use the procedure below to assign complex ambiguity tags to all words in the corpus, based on whether a word is a non-fringe variation nucleus and thus flagged as a potential error by the variation n-gram method (choice 1), or is not a nucleus (choice 2).

1. Every word which is a variation word (nucleus of a non-fringe variation) or type-identical to a variation word is assigned:
   (a) a complex tag reflecting the ambiguity class of all relevant ambiguities in the non-fringe variation nuclei; or
   (b) a simple tag reflecting no ambiguity, if the tag is irrelevant.
2. Based on their relevant unigram tags, non-variation words are assigned:
   (a) a complex tag, if the word's ambiguity tag also appears as a variation ambiguity; or
   (b) a simple tag, otherwise.

Variation words (choice 1) We start with variation nuclei because these are the potential errors we wish to correct. An example of choice 1a is *ago*, which varies between IN and RB as a nucleus, and so receives the tag <IN/RB,IN> when it resolves to IN and <IN/RB,RB> when it resolves to RB.

The choices are based on relevance, though; instead of simply assigning all tags occurring in an ambiguity to an ambiguity class, we filter out ambiguities which we deem *irrelevant*. Similar to Brill and Pop (1999) and Schmid (1997), we do this by examining the variation unigrams and removing tags which occur less than 0.01 of the time for a word and less than 10 times overall. This eliminates variations like ,/DT where DT appears 4210 times for *an*, but the comma tag appears only once. Doing this means that *an* can now be grouped with other unambiguous determiners (DT). In addition to removing some erroneous classes, we gain generality and avoid data sparseness by using fewer ambiguity classes.

This pruning also means that some variation words will receive tags which are not part of a variation, which is when choice 1b is selected. For instance, if the class is IN/RB and the current tag is JJ, it gets JJ instead of <IN/RB,JJ> because a word varying between IN and RB should not resolve to JJ. This situation also arises because we are deriving the ambiguity tags only from the non-fringe nuclei but are additionally assigning them to type-identical words in the corpus. Words involved in a variation may elsewhere have tags never involved in a variation. For example, *Advertisers* occurs as a non-fringe nucleus varying between NNP (proper noun) and NNPS (plural proper noun). In non-variation positions, it appears as a plural common noun (NNS), which we tag as NNS because NNS is not relevant to the variation (NNP/NNPS) we wish to distinguish.

One more note is needed to explain how we handled the vertical slashes used in the Penn Treebank annotation. Vertical slashes represent uncertainty between two tags—e.g., JJ|VBN means the annotator could not decide between JJ and VBN (past participle). Variation between JJ, VBN, and JJ|VBN is simply variation between JJ and VBN, and we represent it by the class JJ/VBN, thereby ensuring that JJ/VBN has more data.

In short, we assign complex ambiguity tags to variation words whenever possible (choice 1a), but because of pruning and because of non-variation tags for a word, we have to assign simple tags to some corpus positions (choice 1b).

Non-variation words (choice 2) In order to have more data for a tag, non-variation words also

take complex ambiguity tags. For words which are not a part of a variation nucleus, we similarly determine relevance and then assign a complex ambiguity tag if the ambiguity is elsewhere involved in a non-fringe nucleus (choice 2a). For instance, even though *join* is never a non-fringe variation nucleus, it gets the tag <VB/VBP,VB> in the first sentence of the treebank because its ambiguity class VB/VBP is represented in the non-fringe nuclei.

On the other hand, we ignore ambiguity classes which have no bearing on correction (choice 2b). For example, *ours* varies between JJ and PRP (personal pronoun), but no non-fringe variation nuclei have this same ambiguity class, so no complex ambiguity tag is assigned. Our treatment of non-variation words increases the amount of relevant data (choice 2a) and still puts all non-varying data together (choice 2b).

Uniform assignment of tags Why do we allow only one ambiguity class per word over the whole corpus? Consider the variation nucleus *traded*: in *publicly traded investments*, *traded* varies between JJ and VBN, but in *contracts traded on*, it varies between VBN and VBD (past tense verb). It seems like it would be useful to keep the JJ/VBN cases separate from the VBD/VBN ones, so that a tagger can learn one set of patterns for JJ/VBN and a different set for VBD/VBN. While that might have its benefits, there are several reasons why restricting words to a single ambiguity class is desirable, i.e., why we assign *traded* the ambiguity class JJ/VBD/VBN in this case.

First, we want to group as many of the word occurrences as possible together into a single class. Using JJ/VBN and VBD/VBN as two separate ambiguity classes would mean that *traded* as VBN lacks a pattern of its own.

Secondly, multiple ambiguity classes for a word can increase the number of possible tags for a word. For example, instead of having only the tag <JJ/VBD/VBN,VBN> for *traded* as VBN, we would have both <JJ/VBN,VBN> and <VBD/VBN,VBN>. With such an increase in the number of tags, data sparseness becomes a problem.

Finally, although we know what the exact ambiguity in question is for a non-fringe nucleus, it is too difficult to go through position by position to guess the correct ambiguity for every other spot. If we encounter a JJ/VBD/VBN word like *followed* tagged as VBN, for example, we cannot know for sure whether this is an instance where JJ/VBN was the decision which had to be made or if VBD/VBN was the difficult choice; keeping only one ambiguity class per word allows us to avoid guessing.

5.3 Results with complex ambiguity tags

Using complex ambiguity tags increases the size of the tagset from 80 tags in the original corpus [7] to 418 tags in the altered tagset, 53 of which are simple (e.g. IN) and 365 of which are complex (e.g. <IN/RB,IN>).

TnT Examining the 300 samples of variation positions from the WSJ corpus for the TnT tagger with complex ambiguity tags, we find that 234 spots are correctly tagged, for a precision of 78.00%. Additionally, we find 73.86% (65/88) precision for tags which have been changed from the original corpus. The 78% precision is a significant improvement both over the original TnT precision of 71.67% ($p = .008$) and the benchmark of 67% ($p = .001$). Perhaps more revealing is the improvement in the precision of the changed tokens, from 58.56% to 73.86%. With 73.86% precision for changed positions, this means that we expect approximately 3968 of the 5373 changes that the tagger makes, out of 21,575 flagged positions, to be correct changes. Thus, the error rate of the corpus will be reduced.

Decision Tree Tagger (DTT) Using complex ambiguity tags with DTT results in an overall precision of 78.33% (235/300) and a precision of 73.56% (64/87) for the changed positions. We improve the overall error correction precision, from 76.33% to 78.33%, and the tagging of changed positions, going from 65.59% to 73.56%.

The results for all four models, plus the baseline, are summarized in figure 1. From these figures, it seems that the solution for error correction lies less in what tagging method is used and more in the information we give each method.

The improvement in changed positions for both TnT and DTT is partly attributable to the fact that both tagging models are making fewer changes. Indeed, training TnT on the original corpus and then testing on the same corpus results in a 97.37% similarity, but a TnT model trained on complex ambiguity tags results in 98.49% similarity with

[7] The number of tags here counts tags with vertical slashes separately.

	Total	Changed
Baseline	67.00%	N/A
TnT	71.67%	58.56% (65/111)
C.A. TnT	78.00%	73.86% (65/88)
DTT	76.33%	65.59% (69/107)
C.A. DTT	78.33%	73.56% (64/87)

Figure 1: Summary of results

the original. DTT sees a parallel overall improvement, from 97.47% to 98.33%. Clearly, then, each complex ambiguity model is a closer fit to the original corpus. Whether this means it is an overall better POS tagging model is an open question.

Remaining issues We have shown that we can improve the annotation of a corpus by using tagging models with complex ambiguity tags, but can we improve even further? To do so, there are several obstacles to overcome.

First, some distinctions cannot be handled by an automated system without semantic or non-local information. As Marquez and Padro (1997) point out, distinctions such as that between JJ and VBN are essentially semantic distinctions without any structural basis. For example, in the phrase *proposed offering*, the reason that *proposed* should be VBN is that it indicates a specific event. Since our method uses no external semantic information, we have no way to know how to correct this.[8]

Other distinctions, such as the one between VBD and VBN, require some form of non-local knowledge in order to disambiguate because it depends on the presence or absence of an auxiliary verb, which can be arbitrarily far away.

Secondly, sometimes the corpus was more often wrong than right for a particular pattern. This can be illustrated by looking at the word *later* in example (11), from the WSJ corpus. In the tagging manual (Santorini, 1990, p. 25), we find the description of *later* as in (12).

(11) Now , 13 years *later* , Mr. Lane has revived his Artist ...

(12) **later** should be tagged as a simple adverb (RB) rather than as a comparative adverb (RBR), unless its meaning is clearly comparative. A useful diagnostic is that the comparative *later* can be preceded by *even* or *still*.

In example (11), along with the fact that this is 13 years later as compared to now (i.e., comparative), one can say *Now, (even) 13 years later, Mr. Lane has revived his Artist ...*, favoring RBR as a tag. But the trigram *years later* , occurs 16 times, 12 as RB and 4 as RBR. Assuming RBR is correct, we clearly have a lot of wrong annotation in the corpus, even though here the corpus is correctly annotated as RBR. As seen in (13), in the context of following CD and NNS, RBR is much less likely for TnT than either RB or JJ.

(13) a. $p(\text{JJ}|\text{CD,NNS}) = .0366$
b. $p(\text{RB}|\text{CD,NNS}) = .0531$
c. $p(\text{RBR}|\text{CD,NNS}) = .0044$

As shown in (14), even when we use complex ambiguity tags, we still find this favoritism for RB because of the overwhelmingly wrong data in the corpus. However, we note that although RB is favored, its next closest competitor is now RBR—not JJ—and RB is no longer favored by as much as it was over RBR. We have more appropriately narrowed down the list of proper tags for this position by using complex ambiguity tags, but because of too much incorrect annotation, we still generate the wrong tag.

(14) a. $p(\text{<JJ/RB/RBR,JJ>}|\text{CD,NNS}) = .0002$
b. $p(\text{<JJ/RB/RBR,RB>}|\text{CD,NNS}) = .0054$
c. $p(\text{<JJ/RB/RBR,RBR>}|\text{CD,NNS}) = .0017$

These issues show that automatic correction must be used with care, but they also highlight particular aspects of this tagset that any POS tagging method will have difficulty overcoming, and the effect of wrong data again serves to illustrate the problem of annotation errors in training data.

6 Summary and Outlook

We have demonstrated the effectiveness of using POS tagging technology to correct a corpus, once an error detection method has identified potentially erroneous corpus positions. We first showed that using a tagger as is provides moderate results, but adapting a tagger to account for problematic tag distinctions in the data—i.e., using complex ambiguity tags—performs much better and

[8] Note that it could be argued that this lack of a structural distinction contributed to the inconsistency among annotators in the first place and thus made error *detection* successful.

reduces the true error rate of a corpus. The distinctions in the tagging model have more of an impact on the precision of correction than the underlying tagging algorithm.

Despite the gain in accuracy, we pointed out that there are still several residual problems which are difficult for any tagging system. Future work will go into automatically sorting the tags so that the difficult disambiguation decisions can be dealt with differently from the easily disambiguated corpus positions. Additionally, we will want to test the method on a variety of corpora and tagging schemes and gauge the impact of correction on POS tagger training and evaluation. We hypothesize that this method will work for any tagset with potentially confusing distinctions between tags, but this is yet to be tested.

The method of adapting a tagging model by using complex ambiguity tags originated from an understanding that the POS tagging process is crucially dependent upon the tagset distinctions. Based on this, the correction work described in this paper can be extended to the general task of POS tagging, as a tagger using complex ambiguity classes is attempting to tackle the difficult distinctions in a corpus. To pursue this line of research, work has to go into defining ambiguity classes for all words in the corpus, instead of focusing on words involved in variations.

Acknowledgments I would like to thank Detmar Meurers for helpful discussion, Stephanie Dickinson for her statistical assistance, and the three anonymous reviewers for their comments.

References

Thorsten Brants. 1996. Estimating Markov model structures. In *Proceedings of ICSLP-96*, pages 893–896, Philadelphia, PA.

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

Eric Brill and Mihai Pop. 1999. Unsupervised learning of disambiguation rules for part of speech tagging. In Kenneth W. Church, editor, *Natural Language Processing Using Very Large Corpora*, pages 27–42. Kluwer Academic Press, Dordrecht.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of ANLP-92*, pages 133–140, Trento, Italy.

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

Markus Dickinson and W. Detmar Meurers. 2003. Detecting errors in part-of-speech annotation. In *Proceedings of EACL-03*, pages 107–114, Budapest, Hungary.

Markus Dickinson. 2005. *Error detection and correction in annotated corpora*. Ph.D. thesis, The Ohio State University.

Eleazar Eskin. 2000. Automatic corpus correction with anomaly detection. In *Proceedings of NAACL-00*, pages 148–153, Seattle, Washington.

Pavel Květŏn and Karel Oliva. 2002. Achieving an almost correct PoS-tagged corpus. In *Text, Speech and Dialogue (TSD 2002)*, pages 19–26, Heidelberg. Springer.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Lluis Marquez and Lluis Padro. 1997. A flexible POS tagger using an automatically acquired language model. In *Proceedings of ACL-97*, pages 238–245, Madrid, Spain.

Lluis Marquez, Lluis Padro, and Horacio Rodriguez. 2000. A machine learning approach to POS tagging. *Machine Learning*, 39(1):59–91.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157.

Karel Oliva. 2001. The possibilities of automatic detection/correction of errors in tagged corpora: a pilot study on a German corpus. In *Text, Speech and Dialogue (TSD 2001)*, pages 39–46. Springer.

Ferran Pla and Antonio Molina. 2004. Improving part-of-speech tagging using lexicalized HMMs. *Natural Language Engineering*, 10(2):167–189.

Beatrice Santorini. 1990. 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.

Helmut Schmid. 1997. Probabilistic part-of-speech tagging using decision trees. In D.H. Jones and H.L. Somers, editors, *New Methods in Language Processing*, pages 154–164. UCL Press, London.

Tylman Ule. 2003. Directed treebank refinement for PCFG parsing. In *Proceedings of TLT 2003*, pages 177–188, Växjö, Sweden.

Hans van Halteren. 2000. The detection of inconsistency in manually tagged text. In Anne Abeillé, Thosten Brants, and Hans Uszkoreit, editors, *Proceedings of LINC-00*, Luxembourg.