Ensemble of Classifiers for Noise Detection in PoS Tagged Corpora

Harald Berthelsen[1,2] and Beáta Megyesi[2]

[1] Telia Promotor, Infovox,
169 02, Solna, Sweden
harald@ling.su.se
http://www.ling.su.se/staff/harald
[2] Computational Linguistics,
Dept. of Linguistics, Stockholm University,
106 91, Stockholm, Sweden
bea@ling.su.se
http://www.ling.su.se/staff/bea

Abstract. In this paper we apply the ensemble approach to the identification of incorrectly annotated items (noise) in a training set. In a controlled experiment, memory-based, decision tree-based and transformation-based classifiers are used as a filter to detect and remove noise deliberately introduced into a manually tagged corpus. The results indicate that the method can be successfully applied to automatically detect errors in a corpus.

1 Introduction

In recent years there has been an increase in the application of several machine learning techniques to Natural Language Processing (NLP). They have proven to be successful in a large number of tasks, such as Part of Speech (PoS) tagging. Part of speech (PoS) taggers are classifiers, used to assign the appropriate, i.e. contextually correct, part of speech label with or without morphological information to every word token in a corpus.

The purpose of this study is to investigate a method to automatically improve the quality of the training data by identifying and correcting the wrongly annotated items in order to achieve higher accuracy of the learning system. If a method along these lines can be shown to work, it could be used to great advantage in the creation of corpora.

Automatic identification and elimination of noise in data sets is a well-known problem in the machine learning community. Several studies show that the elimination of noise in the training set results in higher accuracy. For example, Quinlan [6] showed that cleansing the training data from mislabelled training instances results in a classifier with significantly higher accuracy. This is illustrated by Brodley and Friedl [3], by adding class noise levels of less than 40% and then removing wrongly labelled items from the training data. The result is higher predictive accuracy compared to classification accuracy achieved without removing the mislabelled instances in the training data.

To improve system performance several methods have been suggested for the automatic recognition of noise in data sets [1]. One of these techniques is based on creating ensembles of classifiers. The advantage of using ensembles of classifiers is that they can improve performance, i.e. errors made by an individual classifier can be removed by voting.

In 1999, Brodley & Friedl developed a general method for removing instances that do not follow the same model as the rest of data. Their method for removing outliers differs from the technique used in e.g. regression analysis in the way that the errors in the class labels are independent of the particular learning model being fit to the data. By using m learning algorithms, they create m classifiers that together act as a filter. A classifier is defined as the output of a learning algorithm, given a set of training examples. An ensemble of classifiers, i.e. a set of classifiers where the decisions of each individual classifier are combined by weighted or non-weighted voting to classify new examples is created. The vote of the ensemble is used to remove or reclassify instances that are assumed to be mislabelled. Brodley & Friedl use the filtered data to train a classifier, and show that the performance is better than for a classifier trained on the unfiltered data. The general procedure is shown in Figure 1 below, see [4].

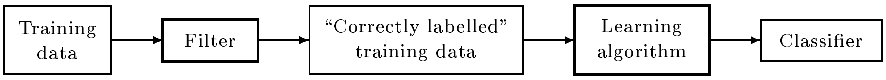

Fig. 1. Method for eliminating incorrectly labelled instances

Thus, by using a set of classifiers formed from part of the training data, it is possible to test whether instances in the remaining part of the training data are mislabelled.

In the domain of NLP, Brill and Wu [2] combined multiple classifiers (based on unigram, n-gram, transformation-based, and maximum entropy tagging) for improved lexical disambiguation. They showed that errors made by different PoS tagger are complementary and hence the taggers can be combined to a single, new tagger for achieving higher accuracy.

2 Method for Noise Detection for PoS Tagging

In this section, we describe an approach, based on Brodley & Friedl [4], for the detection of noise in a training corpus. First, we build classifiers, based on different learning algorithms for varying levels of inserted noise to the training set to act as a filter. Next, we filter the training data using standard ensemble vote procedure: consensus and majority vote filters.

[1] See [3] for a summary of these methods.

2.1 Learning Algorithms

We used three widespread and successful learning algorithms in the experiment: nearest neighbour, decision tree induction, and transformation-based learning. The implementations used are TiMBL [5] for nearest neighbour and decision trees. We choose this system because it has been used in the past with good results to develop PoS-taggers (see [8]). TBL is represented by the PoS-tagger developed by Eric Brill [1].

The *memory-based learning* (MBL) algorithm implemented in the TiMBL system is an example of the k-nearest neighbour algorithm. Features of an instance are represented as elements in a vector. Training consists of storing the selected training instances in memory, and an instance to be classified is compared to all the stored instances. An information gain measure is used to determine the relative weights of different features. The TiMBL system also includes an implementation of *decision tree* induction, called ig-tree. Here, as in MBL, an instance is represented by a vector, where the elements are the different features of the instance. Information gain is used to determine, at each node in the tree, which feature should be used to create new branches. Each of the two algorithms is used with one and three words of context to create four classifiers.

Transformation-based Error-driven Learning (TBL), developed by Eric Brill [1] is based on transformations or rules, and learns by detecting errors. Roughly, the TBL begins with an unannotated text as input which passes through the 'initial state annotator'. It assigns tags to the input in a heuristic fashion. The output of the initial state annotator is a temporary corpus, which is then compared to a goal corpus, i.e. the correctly annotated training corpus. For each time the temporary corpus is passed through the learner, the learner produces one new rule, the single rule that improves the annotation the most compared with the goal corpus. It replaces the temporary corpus with the analysis that results when this rule is applied to it. By this process the learner produces an ordered list of rules.

2.2 Training and Test Data

The corpus used in this study is the SUSANNE corpus[2] that comprises a 152,332 tagged tokens subset of the Brown Corpus of American English consisting of different text types annotated with PoS tags only. Other morpho-syntactical features, that belong in the SUSANNE annotation scheme [7], have been removed. The corpus was randomly divided into two approximately equal parts; one for training and one for test in order to get different text types both for training and test. The training corpus includes 3598 sentences including 76221 words while the test corpus consists of 3596 sentences including 76111 words.

2.3 Adding Noise to the Data

In order to allow for a controlled simulation of the effects of noise in the data, the following procedure was adopted; Words belonging to four ambiguity classes,

[2] The SUSANNE corpus is freely available at ftp://ftp.cogsci.ed.ac.uk/pub/susanne/

i.e words that may be classified in two different ways depending on context, were selected as candidates for noise insertion. The ambiguity classes selected were (1) noun-verb, (2) adverb-preposition, (3) adjective-noun and (4), adjective-adverb. These classes are all known to be error prone in manual tagging and automatic classification alike. A lexicon was built out of the entire corpus, and used to identify all individual words belonging to the four ambiguity classes. A total of 10978 such cases were found in the corpus, 5445 in the training set and 5533 in the test set. In the noise insertion procedure, a percentage of the words belonging to the selected ambiguity classes had their tags altered. In one particular case, for example, the word 'answer' was found in the corpus, with the PoS tag VB, for verb. This word is of course ambiguous between a noun reading and a verb reading, so it belongs in one of the selected ambiguity classes. A random number (between 1 and 100) was drawn and if this number was less than or equal to the noise level set, the PoS tag for this word was changed to NN, the other possible tag in this ambiguity class. In this way the original training and test sets were converted to five new training sets and five new test sets with different levels of noise, from 0 through 40. The noise level is thus the percentage of words, belonging to the selected ambiguity classes, that were actually changed.

2.4 Constructing classifiers

First, the original training corpus and the four training corpora with varying levels of inserted noise were used as training data for five classifiers, where MBL and ig-tree was used in two each, with respectively one and three words as right and left context. The fifth classifier used was Brill's tagger. The ambiguity class for the current word was retrieved from a lexicon of the entire corpus.

The classification task, then, consisted in assigning one PoS tag to each word, based on the ambiguity class of the word and the tags of the surrounding words.

Individual classifier precision In the basic classification task, the accuracy of the classifiers was computed on the entire test set. The outcome is shown in the second column of table 1. The five classifiers perform at very much the same level when assigning a part-of-speech label to each word in the test set without any introduced noise in the training data. MBL, with only one word of left and right context, turns out to be the best, but the difference is small and hardly significant.

However, it is interesting to compare the error rates not on the entire test set, but on only the words belonging to the ambiguity classes selected as candidates for noise insertion. The accuracy of the five classifiers when measured on these words is shown in the third column of table 1. It is immediately obvious that the accuracy is much lower on this subset than on the entire test set. This reflects the fact that these words belong to difficult ambiguity classes, the very reason they were selected for this experiment.

Table 1. Accuracy of the five classifiers, trained and tested on original data. Accuracy on the entire test set (ACCURACY 1), accuracy on the selected ambiguity classes (ACCURACY 2).

CLASSIFIER	ACCURACY 1	ACCURACY 2
MBL, 1-W CONTEXT	97.93 %	90.31 %
IG-TREE, 1-W CONTEXT	97.88 %	90.04 %
MBL, 3-W CONTEXT	97.88 %	89.93 %
IG-TREE, 3-W CONTEXT	97.85 %	89.75 %
TBL	97.87 %	90.11 %
AVERAGE	97.88 %	90.03 %

2.5 Filtering

The second part of this study involves the standard ensemble voting procedure to finally classify an instance.

As we showed in table 1, the difference in accuracy over the entire test set is minimal between the five classifiers. For this reason, assigning different voting weights to the classifiers was deemed inappropriate. Instead, we used two types of non-weighted filters: majority vote filter and consensus filter. A majority vote filter annotates an item as mislabelled if more than 50% of the five individual classifiers agree on assigning a tag different from the original. A consensus filter, on the other hand, is used when all individual classifiers agree to classify an instance as the target label. In case all classifiers disagree, the decision of MBL (one-word context) algorithm is chosen, since this algorithm has the highest overall accuracy (see table 1).

Voting results for the different filters If the filtering is completely successful the tag given by the voting should always be the correct tag and never the erroneous tag assigned by the noise insertion procedure. Table 2 shows the extent to which the filtering is successful, using consensus or majority vote filters.

As table 2 shows, there is a trade off between the removal of noise and the elimination of correct tags. The majority vote filter succeeds better in choosing the correct class among the possible classes but it also removes the correct tag in many cases. On the other hand, the consensus filter takes up a more cautious attitude since it does not remove as many correct tags, but instead removes less noise, compared to the majority vote filter. At a lower noise level, consensus vote can perhaps be said to perform better than majority vote. This trade off could possibly be exploited if one has a general idea of the error level in a corpus.

3 Conclusions and Future Directions

In this small study, we have applied a method for the automatic detection of noise in a part-of-speech tagged training corpus. The results show that a filter

Table 2. Filtering by consensus (CON) and majority (MAJ) vote filters

	NOISE LEVEL	NOISE REMOVED: NO.OF CASES	NOISE REMOVED: PER CENT OF INSERTED NOISE	CORRECT TAGS REMOVED: NO. OF CASES	CORRECT TAGS REMOVED: PER CENT OF ALL CASES
	0	0	–	209	3.8 %
C	10	407	75.9 %	190	3.4 %
O	20	749	68.0 %	183	3.3 %
N	30	842	52.7 %	179	3.2 %
	40	787	36.1 %	272	4.9 %
	0	0	–	533	9.6 %
M	10	474	88.4 %	590	10.7 %
A	20	937	85.1 %	674	12.2 %
J	30	1266	79.3 %	800	14.4 %
	40	1459	66.9 %	982	17.7 %

can successfully be created, by combining classifiers based on different learning algorithms, using standard ensemble vote procedures.

Without performing the final step of Brodley & Friedl, the training of a classifier on the filtered data, the filter can be used as an aid in identifying and correcting misclassified data. This method can be put to use in the development of large corpora. A boot-strapping procedure is envisaged, where a PoS-tagger is trained on a smaller corpus, the tagger is used to tag new text, and the filter is applied to identify errors. In this iterative way, larger corpora and better PoS-taggers can be constructed.

References

1. Brill, E.: Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of speech Tagging. In Computational Linguistics. 21:4.(1995)
2. Brill, E. & Wy, Y.: Classifier Combination for Improved Lexical Disambiguation. In Proceedings of COLING-ACL'98, 1, pp. 191-195. (1998).
3. Brodley, C. E., & Friedl, M. A.: Identifying and eliminating mislabeled training instances. In Proceedings of the Thirteenth National Conference on Artificial Intellience, pp. 799-805 Portland, OR. AAAI Press. (1996).
4. Brodley, C. E., Friedl, M. A.: Identifying Mislabeled Training Data. Journal of Artificial Intelligence Research, 11, pp. 131 - 167. (1999).
5. Daelemans, W., Zavrel, J., Sloot, K, Bosch, A.: TiMBL: Tilburg Memory Based Learner. Reference Guide. ILK Technical Report - ILK 99-01. (1999).
6. Quinlan, J. R.: Induction of decision trees. Machine Learning, 1 (1), 81-106. (1986).
7. Sampson, G.: English for the Computer. Oxford University Press. (1995).
8. Zavrel, J and Daelemans, W.: Recent Advances in Memory-Based Part-of-Speech Tagging. VI Simposio Internacional de Comunicacion Social, Santiago de Cuba, pp. 590-597, 1999. ILK-9903. (1999).