Detecting Errors in Corpora Using Support Vector Machines

Tetsuji Nakagawa* and **Yuji Matsumoto**
Graduate School of Information Science
Nara Institute of Science and Technology
8916−5 Takayama, Ikoma, Nara 630−0101, Japan
nakagawa378@oki.com, matsu@is.aist-nara.ac.jp

Abstract

While the corpus-based research relies on human annotated corpora, it is often said that a non-negligible amount of errors remain even in frequently used corpora such as Penn Treebank. Detection of errors in annotated corpora is important for corpus-based natural language processing. In this paper, we propose a method to detect errors in corpora using support vector machines (SVMs). This method is based on the idea of extracting exceptional elements that violate consistency. We propose a method of using SVMs to assign a weight to each element and to find errors in a POS tagged corpus. We apply the method to English and Japanese POS-tagged corpora and achieve high precision in detecting errors.

1 Introduction

Corpora are widely used in natural language processing today. For example, many statistical part-of-speech (POS) taggers have been developed and they use corpora as the training data to obtain statistical information or rules (Brill, 1995; Ratnaparkhi, 1996). For natural language processing systems based on a corpus, the quantity and quality of the corpus affect their performance. In general, corpora are annotated by hand, and therefore are error-prone. These errors are problematic for corpus-based systems. The errors become false training examples and deteriorate the performance of the systems. Furthermore, incorrect instances may be used as testing examples and prevent the accurate measurement of performance. Many studies and improvements have been conducted for POS tagging, and major methods of POS tagging achieve an accuracy of 96–97% on the Penn Treebank WSJ corpus, but obtaining higher accuracies is difficult (Ratnaparkhi, 1996). It is mentioned that the limitation is largely caused by inconsistencies in the corpus (Ratnaparkhi, 1996; Padró and Màrquez, 1998; van Halteren et al., 2001). Therefore, correcting the errors in a corpus and improving its quality is important. However, to find and correct errors in corpora by hand is costly, since the size of corpora is usually very large. Hence, automatic detection of errors in corpora is necessary.

One of the approaches for corpus error detection is use of machine learning techniques (Abney et al., 1999; Matsumoto and Yamashita, 2000; Ma et al., 2001). These methods regard difficult elements for a learning model (boosting or neural networks) to learn as corpus errors. Abney et al. (1999) studied corpus error detection using boosting. Boosting assigns weights to training examples, and the weights are large for the examples that are difficult to classify. Mislabeled examples caused by annotators tend to be difficult examples to classify and these authors conducted error detection of POS tags and PP attachment information in a corpus by extracting examples with a large weight.

Some probabilistic approaches for corpus error detection have also been studied (Eskin, 2000; Murata et al., 2000). Eskin (2000) conducted corpus error detection using anomaly detection. He supposed that all the elements in a corpus are generated by a mixture model consisting of two distributions, a majority distribution (typically a structured distribution) and an anomalous distribution (a uniform random distribution), and erroneous elements are generated by the anomalous distribution. For each element in a corpus, the likelihood of the mixed

* Presently with Service Media Laboratory, Corporate Research and Development Center, Oki Electric Industry Co.,Ltd.

model is calculated in both cases when the element is generated from the majority distribution and from the anomalous one. The element is detected as an error if the likelihood in the latter case is large enough.

In this paper, we focus on detection of errors in corpora annotated with POS tags, and propose a method for corpus error detection using support vector machines (SVMs). SVMs are one of machine learning models and applied to many natural language processing tasks with success recently. In the next section, we explain a method to use SVMs for corpus error detection.

2 Corpus Error Detection Using Support Vector Machines

Training data for corpus error detection is usually not available, so we have to solve it as an unsupervised learning problem. We consider in the following way: in general, a corpus is built according to a set of guidelines, thus it should be consistent. If there is an exceptional element in the corpus that jeopardizes consistency, it is likely to be an error. Therefore, corpus error detection can be conducted by detecting exceptional elements that causes inconsistency.

While this is a simple and straightforward approach and any machine learning method is applicable to this task, we will use SVMs as the learning algorithm in the settings described in Section 2.2. The advantage of using SVMs in this setting is the following: In our setting, each position in the annotated corpus receives a weight according to the SVM algorithm and these weights can be used as the confidence level of erroneous examples. By effectively using those weights the inspection of the erroneous parts can be undertaken in the order of the confidence level, so that an efficient browsing of corpus becomes possible. We believe this is a particular advantage of our method compared with the methods that use other machine learning methods.

2.1 Support Vector Machines

Support Vector Machines (SVMs) are a supervised machine learning algorithm for binary classification (Vapnik, 1998). Given l training examples of feature vector $\mathbf{x}_i \in \mathbf{R}^L$ with label $y_i \in \{+1, -1\}$, SVMs map them into a high dimensional space by a nonlinear function $\mathbf{\Phi}(\mathbf{x})$ and linearly separate them. The optimal hyperplane to separate them is found by solving the following quadratic programming problem:

$$\begin{aligned} \underset{\alpha_1,\ldots,\alpha_l}{\text{minimize}} \quad & \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{l}\alpha_i, \\ \text{subject to} \quad & 0 \le \alpha_i \le C \qquad (1 \le i \le l), \\ & \sum_{i=1}^{l} \alpha_i y_i = 0, \end{aligned}$$

where the function $K(\mathbf{x}_i, \mathbf{x}_j)$ is the inner product of the nonlinear function ($K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j)$) called a kernel function, and the constant C controls the training errors and becomes the upper bound of α_i. Given a test example $\mathbf{x}$, its label y is decided by summing the inner products of the test example and the training examples weighted by α_i:

$$y \;=\; \mathrm{sgn}\Big(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\Big),$$

where b is a threshold value. Thus, SVMs assign a weight α_i to each training example. The weights are large for examples that are hard for SVMs to classify, that is, exceptional examples in training data have a large weight. We conduct corpus error detection using the weights.

To detect exceptional examples in a corpus annotated with POS tags, we first construct an SVM model for POS tagging using all the elements in a corpus as the training examples. Note that each example corresponds to a word in the corpus. Then SVMs assign weights to the examples, and large weights are assigned to difficult examples. Finally, we extract examples with a large weight greater than or equal to a threshold value θ_α. In the next subsection, we describe how to construct an SVM model for POS tagging.

2.2 Revision Learning for POS tagging

We use a revision learning method (Nakagawa et al., 2002) for POS tagging with SVMs[1]. This method creates training examples of SVMs with

[1]The well known one-versus-rest method (Allwein et al., 2000) can be also used for POS tagging with SVMs, but it has large computational cost and cannot handle segmentation of words directly that is necessary for Japanese morphological analysis.

binary labels for each POS tag class using a stochastic model (e.g. n-gram) as follows: each word in a corpus becomes a positive example of its POS tag class. We then build a simple stochastic POS tagger based on n-gram (POS bigram or trigram) model, and words in the corpus that the stochastic model failed to tag with a correct part-of-speech are collected as negative examples of the incorrect POS tag class. In such way, revision learning makes a model of SVMs to revise outputs of the stochastic model.

For example, assume that for a sentence:

```
"11/CD million/CD yen/NNS are/VBP paid/VBN",
```

a stochastic model tags incorrectly:

```
"11/CD million/CD yen/NN are/VBP paid/VBN".
```

In this case, the following training examples are created for SVMs (each line corresponds to an example):

```
<Class (Label)>  <Feature Vector>
  CD  (+1)  (word:11, word-1:BOS, ...)
  CD  (+1)  (word:million, word-1:11, ...)
  NN  (-1)  (word:yen, word-1:million, ...)
  NNS (+1)  (word:yen, word-1:million, ...)
  VBP (+1)  (word:are, word-1:yen, ...)
  VBN (+1)  (word:paid, word-1:are, ...)
```

Thus, the positive and negative examples are created for each class (POS tag), and a model of SVMs is trained for each class using the training examples.

In English POS tagging, for each word w in the tagged corpus, we use the following features for SVMs:

1. the POS tags and the lexical forms of the two words preceding w;
2. the POS tags and the lexical forms of the two words succeeding w;
3. the lexical form of w and the prefixes and suffixes of up to four characters, the existence of numerals, capital letters and hyphens in w.

Japanese morphological analysis can be conducted with revision learning almost in the same way as English POS tagging, and we use the following features for a morpheme μ:

1. the POS tags, the lexical forms and the inflection forms of the two morphemes preceding μ;
2. the POS tags and the lexical forms of the two morphemes succeeding μ;
3. the lexical form and the inflection form of μ.

2.3 Extraction of Inconsistencies

So far, we discussed how to detect exceptional elements in a corpus. However, it is insufficient and inconvenient for corpus error detection, because an exceptional element is not always an error, that is, an exceptional element may be a correct or an incorrect exceptional element. Furthermore, it is often difficult to judge whether it is a true error or not when only the exceptional element is shown. To solve these problems, we extract not only an exceptional example but also another similar example that is inconsistent with the exceptional example. If the exceptional example is correct, the second example is likely to be an error, and vice versa.

We assume that an inconsistency occurs when two examples have similar features but have opposite labels. The similarity between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ on SVMs is measured by the following distance:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\|\mathbf{\Phi}(\mathbf{x}_i) - \mathbf{\Phi}(\mathbf{x}_j)\|^2}, \\ = \sqrt{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)}.$$

We can extract inconsistencies from a corpus as follows: given an example $\mathbf{x}$ which was detected as an exceptional example (following the proposal in the previous subsection), we extract an example $\mathbf{z}$ with the smallest values of the distance $d(\mathbf{x}, \mathbf{z})$ from the examples whose label is different from $\mathbf{x}$. Intuitively, $\mathbf{z}$ is a closest opposite example to $\mathbf{x}$ in the SVMs' higher dimensional space and may be a cause for $\mathbf{x}$ to be attached a large weight.

3 Experiments

We perform experiments of corpus error detection using the Penn Treebank WSJ corpus (in English), the RWCP corpus (in Japanese) and the Kyoto University Corpus (in Japanese). In the following experiments, we use SVMs with second order polynomial kernel, and the upper bound value C is set to 1.

Table 1: Examples of Correctly Detected Errors and Incorrectly Detected Errors in the WSJ Corpus

Correctly Detected Errors	
pay about 11 million yen/NNS ($ 77,000	budgeted about 11 million yen/NN ($ 77,500
, president and chief/JJ executive officer of	named president and chief/NN executive officer
for its fiscal first quarter ended/VBN Sept. 30	its first quarter ended/VBD Sept. 30 was
Incorrectly Detected Errors	
EOS 3/LS . EOS Send your child to	Nov. 1-Dec . EOS 3/CD . EOS

3.1 Experiments on the Penn Treebank WSJ Corpus (English)

Experiments are performed on the Penn Treebank WSJ corpus, which consists of 53,113 sentences (1,284,792 tokens).

We create models of SVMs for POS tagging using the corpus with revision learning. The distribution of the obtained weights α_i are shown in Figure 1. The values of α_i concentrate near the lower bound zero and the upper bound C. The examples with α_i near the upper bound seem to be exceptional. Therefore, we regarded the examples with $\alpha_i \geq 0.5$ as exceptional examples (i.e. $\theta_\alpha = 0.5$). As a result, 1,740 elements were detected as errors. We implemented a browsing tool for corpus error detection with HTML (see Figure 2). A detected inconsistency pair is displayed in the lower part of the screen. We examined by hand whether the detected errors are true errors or not for the first 200 elements in the corpus from the detected 1,740 elements, and 199 were actual errors and 1 was not. The precision (the ratio of correctly detected errors for all of the detected errors) was 99.5%. Examples of correctly detected errors and incorrectly detected errors from the corpus are shown in Table 1. The underlined words were detected as errors. To judge whether they are true errors or not is easy by comparing the pair of examples that contradict each other.

To examine the recall (the ratio of correctly detected errors for all of the existing actual errors in corpora), we conduct another experiments on an artificial data. We made the artificial data by randomly changing the POS tags of randomly selected ambiguous tokens in the WSJ corpus. The tags of 12,848 tokens (1% for the whole corpus) are changed, and the results

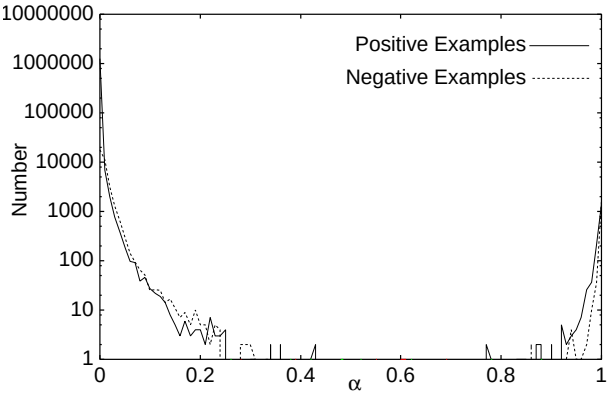

Figure 1: Distribution of the Value α on the WSJ Corpus

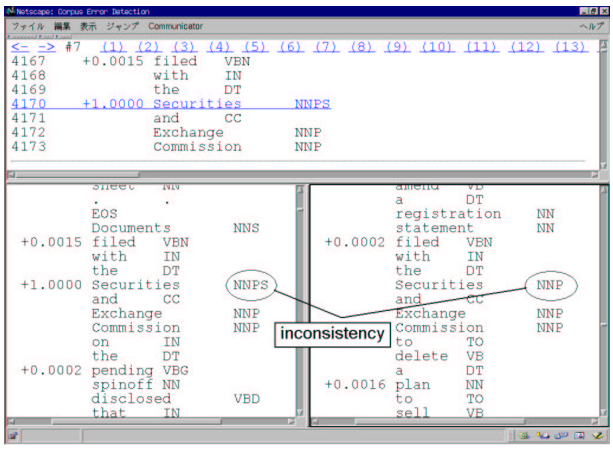

Figure 2: A Tool for Corpus Error Detection

are shown in Table 2 for various values of θ_α[2]. For the smaller threshold θ_α, the larger recall were obtained, but the value is not high.

[2]Precisions cannot be calculated automatically because actual errors as well as the mixed errors are also detected.

Table 2: Recall for the Artificial Data

θ_α	# of Correctly Detected Errors	Recall
1.0	607	4.7%
0.5	1520	11.8%
0.2	1555	12.1%
0.1	1749	13.6%
0.05	2381	18.5%

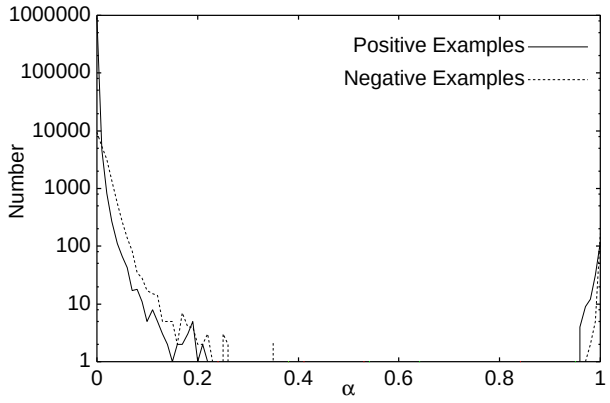

Figure 3: Distribution of the Value α on the RWCP Corpus

3.2 Experiments on the RWCP Corpus (Japanese)

We use the RWCP corpus, which consists of 35,743 sentences (921,946 morphemes).

The distribution of the weights α_i are shown in Figure 3. The distribution of α_i shows the same tendency as in the case of the WSJ corpus.

We conducted corpus error detection for various values of θ_α, and examined by hand whether the detected errors are true errors or not. The results are shown in Table 3, where the correctly detected errors are distinguished into two types, one type is errors of word segmentation and the other is errors of POS tagging, since Japanese has two kinds of ambiguities, word segmentation and POS tagging. Precision of more than 80% are obtained, and the number of POS tag errors is larger than that of segmentation errors.

Examples of correctly detected errors and incorrectly detected errors from the corpus are shown in Table 4. The underlined morphemes were detected as errors. In the examples of correctly detected errors, both segmentation errors (upper) and POS tag errors (lower) are detected. On the other hand, the examples of incorrectly detected errors show the limitations of our method. We use the two morphemes on either side of the current morpheme as features for SVMs. In the examples, the two morphemes on either side are the same and only the POS tag of the current morpheme is different, so that SVMs cannot distinguish them and regard them as errors (inconsistency).

3.3 Experiments on the Kyoto University Corpus (Japanese)

Experiments are performed on a portion of the Kyoto University corpus version 2.0, consisting of the articles of January 1, and from January 3 to January 9 (total of 9,204 sentences, 229,816 morphemes). We set the value of θ_α to 0.5.

By repeating corpus error detection and correction of the detected errors by hand, new errors that are not detected previously may be detected. To examine this, we repeated corpus error detection and correction by hand. Table 5 shows the result. All the detected errors in all rounds were true errors, that is, the precision was 100%. Applying the corpus error detection repeatedly, the number of detected errors decrease rapidly, and no errors are detected in the fourth round. In short, even if we repeat corpus error detection with feedback, few new errors were detected in this experiment.

4 Discussion

Compared to conventional probabilistic approaches for corpus error detection, although precise comparison is difficult, our approach achieved relatively high precision. Using a probabilistic approach, Murata et al. (2000) detected errors of morphemes in a corpus with a precision of 70−80%, and Eskin (2000) detected errors with a precision of 69%, but our approach achieved more than 80%. The probabilistic methods cannot handle infrequent events or compare events with similar probabilities, since the probabilities cannot be calculated or compared with enough confidence, but our method can handle such infrequent events.

SVMs are similar to boosting, and our approach uses the weights attached by SVMs in a similar manner to what Abney et al. (1999) studied. However, we introduced a post-processing step to extract inconsistent similar

Table 3: Number of Detected Errors on the RWCP Corpus

θ_α	Correct Detection	(Segmentation Error / POS Tag Error)	Incorrect Detection	Precision
1.0	110	(30 / 80)	8	93.2%
0.5	165	(43 / 122)	11	93.8%
0.2	171	(45 / 126)	12	93.4%
0.1	188	(51 / 137)	31	85.8%
0.05	300	(73 / 227)	73	80.4%

Table 4: Examples of Correctly Detected Errors and Incorrectly Detected Errors in the RWCP Corpus

Correctly Detected Errors	
用 |多目的/Common-Noun| 車	用 | 多/Prefix|目的/Common-Noun| 車
大阪 | 市 |中央/ProperNoun-Place| 区 | の	大阪 | 市 |中央/Common-Noun| 区 | の
Incorrectly Detected Errors	
容疑 | 者 |と/Coord-Particle| 二 | 人 | の | 間	容疑 | 者 |と/Case-Particle| 二 | 人 | で

Table 5: Number of Detected Errors on the Kyoto University Corpus for Repeated Experiment

Round	1	2	3	4
Correct Detection	85	11	2	0
(Segmentation Error)	(21)	(2)	(0)	(0)
(POS Tag Error)	(64)	(9)	(2)	(0)
Incorrect Detection	0	0	0	0
Total	85	11	2	0

examples, and this improved the precision of detection and usability. Ma et al. (2001) studied corpus error detection by finding conflicting elements using min-max modular neural networks. Compared to their method, our method is useful in the point that the detected errors can be sorted by the attached weights, because human can check more likely elements first.

In the experiment, our method had a high precision but a low recall. The value will be controlled by tuning the features for SVMs as well as the threshold value θ_α, and detecting more errors in a corpus remains as future work.

5 Conclusion

In this paper, we proposed a method for corpus error detection using SVMs. This method can extract inconsistencies in corpora. We achieved precision of 80–100% and showed that many annotation errors exist in widely used corpora. The performance seems to be high enough for practical use in corpus refinement.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting Applied to Tagging and PP Attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 38–45.

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. 2000. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. In *Proceedings of 17th International Conference on Machine Learning*, pages 9–16.

Eric Brill. 1995. Transformation-Based Error-

Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565.

Eleazar Eskin. 2000. Detecting Errors within a Corpus using Anomaly Detection. In *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics*, pages 148–153.

Qing Ma, Bao-Liang Lu, Masaki Murata, Michinori Ichikawa, and Hitoshi Isahara. 2001. On-Line Error Detection of Annotated Corpus Using Modular Neural Networks. In *Proceedings of International Conference on Artificial Neural Networks (ICANN 2001)*, pages 1185–1192.

Yuji Matsumoto and Tatsuo Yamashita. 2000. Using Machine Learning Methods to Improve Quality of Tagged Corpora and Learning Models. In *Proceedings of the Second International Conference on Language Resource and Evaluation*, pages 11–16.

Masaki Murata, Masao Utiyama, Kiyotaka Uchimoto, Qing Ma, and Hitoshi Isahara. 2000. Corpus Error Detection and Correction Using the Decision-List and Example-Based Methods. In *Information Processing Society of Japan SIG Notes, Natural Language No.136*, pages 49–56. (in Japanese).

Tetsuji Nakagawa, Taku Kudo, and Yuji Matsumoto. 2002. Revision Learning and its Application to Part-of-Speech Tagging. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. (to appear).

Lluís Padró and Lluís Màrquez. 1998. On the Evaluation and Comparison of Taggers: the Effect of Noise in Testing Corpora. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 997–1002.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. 2001. Improving Accuracy in Wordclass Tagging through Combination of Machine Learning Systems. *Computational Linguistics*, 27(2):199–230.

Vladimir Vapnik. 1998. *Statistical Learning Theory*. Springer.