Detection of errors in training data by using a decision list and Adaboost

Hiroyuki Shinnou

Ibaraki University, Department of Systems Engineering
4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, JAPAN
shinnou@dse.ibaraki.ac.jp

Abstract

The inductive learning is effective for a variety of natural language processing problems. However, it needs expensive training data. The quality of learned rules often depends on the quality of training data used. In this paper, we propose a method to detect errors in training data automatically to improve the quality of the training data. We consider a Japanese word segmentation problem, in which training data means the corpus including word boundary signs. We use two clues to detect errors. One is a property of a decision list, and the other is the nature of Adaboost. In experiment, we provided 100 doubtful instances from the Kyoto Univeristy Corpus, which is the most trustworthy Japanse tagged corpus. Most of them had tags which we could not easily determine to be correct or incorrect. Moreover 21 instances were actually errors.

1 Introduction

In natural language processing, it is effective to convert problems to classification problems, and to solve them by an inductive learning method. In this strategy, the inductive learning needs a large amount of quality training data. However, training data are generally compiled by hand, and are thus expensive to construct. As a result, the quality of learned rules often depends on the quality of training data used. Therefore, improving the quality of training data is effective to improve the learned rules.

In this paper, we propose a method to detect errors in training data automatically to improve the quality of the training data. Training data often has errors because it is compiled manually. Therefore detection of errors in training data and correction of them improve quality of training data. In this paper, we consider a Japanese word segmentation problem, in which training data means the corpus including word boundary signs, which we call the word segmented corpus. Thus our method is regarded as the method to detected segmentation errors in the word segmented corpus.

We use two clues to detect errors in the training corpus. The first clue is that the higher rank the evidence in the decision list[Yarowsky, 1994] is, the higher precision it has. First we learn the decision list for the word segmentation task through training data. Next using the obtained decision list we conduct word segmentation for training data. This property means that the judgment of an instance through the high ranked evidence is almost correct. Therefore we presume that the tag assigned to the instance is incorrect if the judgment made based on the high ranked evidence is incorrect.

The second clue is the nature of Adaboost. Adaboost is a method to construct a strong learner by combining weak learners. Adaboost builds the first classifier, which is the word segmentation rules, through training data. Then it uses this classifier to determine each instance of training data. Then we pick up the misjudged instances. Adaboost gives added weight to these instances to re-construct training data, and learns the second classifier through it. This process is iterated T times, so that Adaboost makes T classifiers. Final judgment is made by a weighted majority vote of T answers from T classifiers. In the iteration, tags of instances which Adaboost cannot learn easily are probably incorrect[Freund and Schapire (translation by Naoki Abe), 1999][Abney *et al.*, 1999]. Therefore, if an instance gains much weight in the boosting process, the tag assigned to this instance is presumed incorrect.

In the experiment, we applied our method to the Kyoto University Corpus, where word boundary tags and sentence structure tags are assigned manually. We sorted instances in the order of possibility of error, and provided 100 instances with the highest ranks. Most of them have tags which we cannot easily determine to be correct or incorrect. There are also 21 instances that have entirely incorrect tags.

2 Word segmentation by the decision list

Let us assume the input sentence s composed of n characters to be $s = c_1 c_2 \cdots c_n$ where c_i is a character. We can conduct word segmentation by assigning a class $+1$ or -1 to b_i which is the position between c_i and c_{i+1}. The class $+1$ or -1 means that the word boundary exists or does not exist respectively. That is, we can regard the word segmentation problem as the classification problem that we assign a class $+1$ or -1 to b_i ($i = 1, 2, \cdots, n-1$).

There are various methods to solve classification problems, and the best method depends on the type of problem. In this paper, we use the decision list[Yarowsky, 1994].

2.1 Construction of the decision list

The decision list is a kind of inductive learning method which learns classification rules from training data. Essentially, it is a table in which evidences are arranged to identify the class in the order of strength of identifying the class. The evidence is the pair which consists an attribute and its value. The class is determined by the evidence, with the highest identifying strength, in the context. In this paper, we call the pair which consists the evidence and the output class (which is a line in the decision list) a *rule*. Thus, the decision list arranges rules in order of reliability.

The decision list is constructed as follows.

Step 1 Set the attributes.

For example, we assume n attributes $att_1, att_2, \cdots, att_n$.

Step 2 Derive the frequency of collocation between the evidence and the class through training data.

Suppose the value of an attribute att of an instance is a and the class of this instance is C. In this case, we add 1 to the frequency of the collocation $((att, a), C)$ between the evidence (att, a) and the class C. This processing is done for all attributes of all instances in the training data.

Step 3 Define the output class for each evidence and its identifying strength.

If the frequency of collocation $((att, a), C)$ is f_C, the class $\hat{C}$ which gives the maximum value of f_C is the output class for this evidence (att, a), and its identifying strength $pw(att, a)$ is defined by the following expression:

$$pw((att, a)) = \log \frac{f_{\hat{C}} + \alpha}{\sum_{C \neq \hat{C}} f_C + \alpha}.$$

In this paper, we set $\alpha = 0.1$[1].

Step 4 Arrange identifying strengths $pw((att, a))$ in order of the dimensions.

We construct the decision list by arranging all collocation $((att, a), C)$ in order of the dimensions of identifying strengths $pw((att, a))$.

2.2 Setting of attributes

The attributes are clues to determine which class to assign to b_i. As the attributes of b_i, we set up the seven attributes shown in Table 1.

We use the nine general character types shown in Table 2. Detailed character types are defined by dividing Hiragana type in Table 2 into the Hiragana character itself.

Table 1: Attributes

Attribute	Value
att_1	$c_{i-1}c_ic_{i+1}$
att_2	$c_ic_{i+1}c_{i+2}$
att_3	$c_{i-1}c_i$
att_4	c_ic_{i+1}
att_5	$c_{i+1}c_{i+2}$
att_6	(general character type of c_i , general character type of c_{i+1})
att_7	(detailed character type of c_i, detailed character type of c_{i+1})

Table 2: General character types

Character type sign	Meaning
hira	hiragana
kata	katakana
suu	kanji number
kan	kanji
num	Arabic number
alp	alphabet
kigo	symbol
smar	small circle or zero
lmar	large circle or zero

3 Detection of word segmentation errors

3.1 Error detection by the decision list

The higher rank the evidence in the decision list has, the higher the precision is. From this property, we expect the judgment based on the high ranked evidence to be correct. Thus, if the judgment is inconsistent, there is high possibility that the focused instance is incorrect. By using this property, we select instances which are inferred to be incorrect.

Our decision list determines whether or not a word boundary exists between two characters. In training data, the correct answer is assigned to an instance, which means the context under the position between two characters. Thus, when we conduct word segmentation by the decision list for training data, we can pick up instances for which our judgments are inconsistent, These inconsistent instances are assigned information of how the evidence used in the judgment was ranked. Therefore instances with high ranks among selected instances are inferred to be incorrect.

3.2 Error detection by Adaboost

Learning techniques that construct a strong learner by combining some weak learners are called boosting methods. Adaboost is the most popular boosting method, and has been theoretically and empirically proved to be effective.

Figure 1 shows the algorithm of Adaboost[Freund and Schapire (translation by Naoki Abe), 1999]. First, Adaboost gives the same weight to each instance in training data. Next the first classifier is learned through this training data, and then it judges a class of each instance in training data. In the next step, Adaboost adds the weight to instances which the first classifier failed to judge, in order to re-construct training data. The second classifier is learned through this reconstructed training data. These steps are iterated T times to make T classifiers. Finally, we decide the class by a weighted

[1] The addition of a small value is an easy and effective way to avoid an unsatisfactory case, as shown in [Yarowsky, 1994].

Given: $(x_1, y_1), \cdots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{1, -1\}$
Initialize $Di(i) = 1/m$
For $t = 1, \cdots, T$

- Train weak learner using distribution D_t
- Get weak hypothesis $h_t : X \rightarrow Y$ with error

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

- Choose $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
- Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & if \quad h_t(x_i) = y_i \\ e^{\alpha_t} & if \quad h_t(x_i) \neq y_i \end{cases}$$

  where Z_t is a normalization factor

Output the final hypothesis:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

Figure 1: Adaboost

majority vote of T answers from T classifiers, where α_t is the weight assigned to the t-th classifier.

In this paper, the classifier learned from training data becomes the decision list and the weight becomes the frequency. In short, we assume that an instance in training data has a weight, and the weight is used as the frequency in the second step in the procedure constructing the decision list. In the algorithm shown in Figure 1, the sum of weights to be normalized is 1. To simplify the calculation, we added the weight to the instance so the smallest weight would be 1. Therefore, at first every instance in training data has the same weight 1. In the next step, the weights of instances which the first decision list judged correctly are not changed, but the weights of instances which it judged wrong are increased.

Adaboost focuses on the hardest instances, as the instances with the highest weight often turn out to be wrong. By using this property, we can detect errors in training data. However, we must first decide the best number T of decision lists by experiments, and finally judge by a weighted majority vote of T judgments from T decision lists. If the final judgment for an instance is inconsistent, the instance is given a very large weight in the next step of boosting. As a result, instances with final judgment failures are assumed to be incorrect.

3.3 Error detection by combining the decision list and Adaboost

We have described two methods to detect errors in training data by using the decision list and Adaboost, respectively. Here we combine them to detect errors more efficiently.

First, for each instance we record how the evidence used to judge its instance was ranked, and we call this record R1. We do not care whether the judgment is true or false. Next we select doubtful instances by the method using Adaboost, and we call these instances R2. Next we assign a rank to every instance in R2 through the record R1, and sort them according to the order of their rank. The sorted instances are arranged in the order of possibility of error. In this paper, we provide 100 instances with the highest ranks.

4 Experiments

We applied our method to the Kyoto University Corpus, where word boundary tags and sentence structure tags are assigned manually. In the Kyoto University Corpus' 38,383 sentences[2], there were 1,635,505 places where we could determine whether a word boundary existed or not. Each place had a context that corresponds to an instance in the training data. As the result, we made 1,635,505 instances.

Using this training data, we made a decision list. Then we determined the classes of every instance by the decision list, and recorded how the evidence used for judgment of the instance had been ranked.

Next we applied Adaboost to the decision list. The results are shown in Figure 2.

In Figure 2 the x-axis shows the repetition of boosting and the y-axis shows the precision. As we can see from this fig-

[2] In this paper we counted the number of sentences not by the number of periods, but by the number of "EOS" signs.

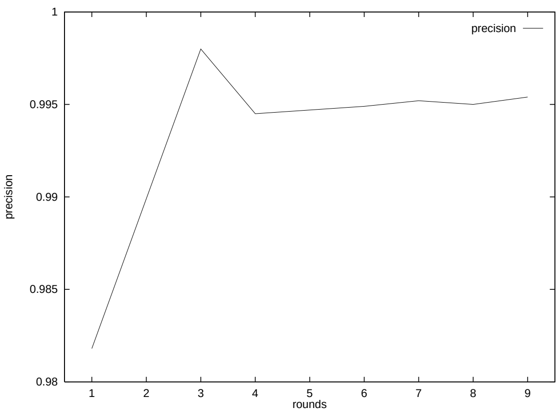

Figure 2: Adaboost of decision lists

ure, the best score was achieved when three decision lists were made by the boosting and the judgment was done by the weighted majority vote of the three decision lists. Thus the number of false judgments is 3,349. Thus these 3,349 instances have a high possibility to be errors.

Next we checked the ranks of these 3,349 instances from the record made by the first decision list. Finally, we provided 100 instances with the highest ranks among them. These instances were presumed to be errors in the training data.

We investigated these 100 instances, and classified them as follows.

(1) Actual errors (21 instances) These instances were actual errors. This type is success of our method.

(2) Collocations (25 instances) In training data, a phrase is regarded as one word. Such phrases are collocations. However, our method often determined that a collocation was not one word, but was segmented into two or more words. As a result, we provided these as errors. It is subtle whether this type is success or failure of our method.

(3) Expressions written in Kana characters (38 instances) In Japanese, phrases of foreign origin are written in Kana characters. It is not clear whether such phrases can be regarded as one word or not. Even training data is not consistent. Thus, our judgments for these phrases often differ from training data. It is also subtle whether this type is success or failure of our method.

(4) Others (16 instances) These instances were our misdetections. That is, tags of these instances in training data were correct. Of course, our method failed to detect these. We note that all these instances occurred in expressions written in Hiragana characters.

Type (1) is regarded as the correct detection, so the precision was 0.21. We take this value as fairly good because training data were carefully compiled by hand. Moreover types (2) and (3) are subtle whether they are correct or not. Therefore we consider that most of our detection (84%) was accurate.

Last we note that there are some cases where two or more incorrect segmentations exists in one word. For example, *'komagatake'* was segmented into *'koma/ga/take'* in training data, and we correctly determined that the segmentaion is *'komagatake'* [3]. Even in this case, we count each segmentation point, which is 2 in this example.

4.1 Effectiveness of use of the decision list

In this paper, we made three decision lists (h_1, h_2, h_3), and used them to determine the class of an instance by a weighted majority vote of three judgments. Through failures of these judgments, we selected 3,349 doubtful instances. Next we sorted these 3,349 instances in the order defined by the decision list h_1.

We then investigated the effectiveness of this order. First, for all 3,349 instances we manually checked whether their classes given in the training data are correct or incorrect. As a result, we found 215 errors [4]. To view the relation between the number of errors and the order defined by the decision list h_1, we investigated the precision p, assuming that we could provide instances with ranks higher than the rank x. In Figure 3, we show the graph which has x as the x-axis and p as the y-axis. This graph shows that the higher rank is, the higher precision is. That is, the order defined by the decision list h_1 is effective.

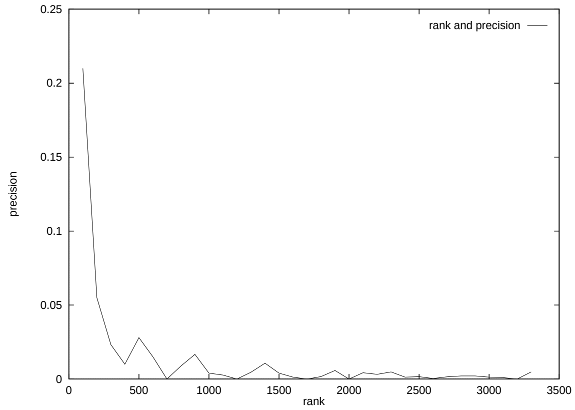

Figure 3: Rank and precision

4.2 Effectiveness of Adaboost

Without using Adaboost, we sorted 29,732 instances of failures of the decision list h_1, in the order defined by h_1 itself, and provided 100 instances with the highest ranks. As a result we found 12 actual errors in these. This number is smaller than the number of actual errors detected by our method. This shows the effectiveness of Adaboost.

5 Discussions

5.1 Use of the first decision list

The decision list h_1 was used to sort 3,349 instances which three decision lists h_1, h_2, h_3 had shown as doubtful errors.

[3] The word *'komagatake'* is a location name, that is one word.

[4] We have already reported these errors to the developers of the Kyoto University Corpus.

However each instance has information of three weights of three decision lists h_1, h_2, h_3. By using these weights, we can sort these instances in the order of the margin. The margin of the instance (x, y) is defined by the following equation.

$$\frac{y \sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}$$

Roughly speaking, the margin corresponds to the distance from the plane which separates positive and negative instances. The smaller the margin is, the subtler the distinction is. Thus we can sort instances in the order of the small margin. This order means the possibility of error. Therefore we may not need to use the decision list h_1.

However, we only used three decision lists. As a result, the margin has only 3 kinds of values, and the highest 1,741 instances have the same margin 0.28. Thus we cannot sort instances in detail. Therefore we used the decision list h_1.

5.2 Use as the general tagging system

We should note that the system which can detect incorrect tags is not always the extremely precise tagging system. There are two reasons. One is that the error detection system does not always give the correct answer for the instance which the error detection system judges to be correct. First of all we need a tagging system, and next we need the system to detect and correct errors in the result produced from the tagging system. The system combining the tagging system and the repairing system may be more precise than the original tagging system. Another reason is the problem of the precision. The combined system is not always better than the original system even if the error detection system is useful.

In this paper, we used JUMAN[5] as the original tagging system. The combination JUMAN and our error detection system was less than JUMAN, because there are 21 right correction but the remaining 79 correction produce 79 new errors. However, most of these 79 instances occur in Kana or Hiragana phrases. By using this property, the combined system may be improved a little.

Whether our error detection system can be used as the general tagging system depends on the precision of the learned decision list. We have conducted some corroborative experiments[Shinnou, 2001], but we have not yet made a system more precise than JUMAN. Our method has clear limitations because it uses only some characters before and after the focused place to judge whether the word boundary exists or not in that place. In order to use our error detection system as a general word segmentation system, we should use also dictionaries.

5.3 Application to other tags

In this paper, we considered only tags of word segmentation, but our method can also be applied to other kinds of tags. To apply our method to other tags, we should note two points. One is to convert the tagging problem to a classification problem. Another is to use the decision list to solve that classification problem. For example, our method can be used directly for homophone problems[Shinnou, 1999]. Assuming that homophones in newspaper articles are written correctly, we generally use such articles as training data to solve homophone problems. However this assumption is not always true. By correcting errors in training data, the system for solving homophone problems can be improved[Ibuki and Nishino, 1998].

In future we will apply our method to other kind of tags.

[5] JUMAN is the Japanese standard morphological analysis system.

6 Conclusions

In this paper, we proposed a method to detect errors in training data automatically to improve the quality of the training data. Our method used two properties to detect errors. One is that the higher rank the evidence in the decision list, the higher precision it has. The other is that Adaboost adds much weight in the boosting process to incorrect instances.

In the experiment, we applied our method to the Kyoto University Corpus. We sorted instances in the order of possibility of error, and provided 100 instances with the highest ranks. Most of them had tags which we could not easily determine to be true or false. Moreover 21 instances were actually errors. This shows the usefulness of our method. In future, we will apply our method to other kind of tags.

References

[Abney *et al.*, 1999] Steven Abney, Robert Schapire, and Yoram Singer. Boosting Applied to Tagging and PP Attachment. In *the joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*,pages 38–45, 1999.

[Freund and Schapire (translation by Naoki Abe), 1999] Yoav Freund and Robert Schapire (translation by Naoki Abe). A short introduction to boosting (in japanese). *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[Ibuki and Nishino, 1998] Jun Ibuki and Fumihito Nishino. Ayamari wo fukumi eru ko-pasu kara no kousei sien you de-ta no seibi (in Japanese). In *The fifth Annual Meeting of The Association for Natural Language Processing*, pages 177–180, 1998.

[Shinnou, 1999] Hiroyuki Shinnou. Detection of Japanese Homophone Errors by a Decision List Including a Written Word as a Default Evidence. In *9th Conference of the European Chapter of the Association for Computational Linguistics (EACL-99)*, pages 180–187, 1999.

[Shinnou, 2001] Hiroyuki Shinnou. Japanese word segmentation by Adaboost using the decision list as the weak learner (in Japanese). *Journal of Natural Language Processing*, 8(2):3–18, 2001.

[Yarowsky, 1994] David Yarowsky. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *32th Annual Meeting of the Association for Computational Linguistics*, pages 88–95, 1994.