Japanese Dependency Analysis Based on Improved SVM and KNN

ZHOU HUIWEI and YANG YAGE and YU TONG and HUANG DEGEN
Department of Computer Science and Engineering
Dalian University of Technology
DaLian , LiaoNing
P.R. CHINA

Abstract: - This paper presents a method of Japanese dependency structure analysis based on improved Support Vector Machine (SVM). Japanese dependency analyzer based on SVM has been proposed and has achieved high accuracy. The efficient way to improve dependency accuracy farther is to increase the training data. However, the increase of training data will bring a great amount of training cost and decrease the parsing efficiency. We delete those samples that are unused or not good to improve the classifier's performance, and then train the reduced training set with SVM to obtain the final classifier. Furthermore, we combine improved SVM with K nearest neighbors(KNN) to improve the performance of dependency analyzer. Experiments using the Kyoto University Corpus show that the method outperforms previous systems as well as the dependency accuracy and the parsing efficiency.

Key-Words: - Japanese dependency analysis, Support Vector Machine(SVM), Improved SVM, Large training set SVM (LSVM), Nearest Neighbor-SVM (NN-SVM), K nearest neighbors(KNN)

1 Introduction

Dependency analysis has been recognized as a basic process in Japanese sentence analysis. And a number of studies have been proposed. Japanese dependency is usually in terms of relationship between phrasal units called bunsetsu segments (hereafter segments).

In recent years, as large-scale tagged corpora have become available, a number of statistical parsing techniques using such tagged corpora have been developed [1] [2] [3] [4]. The previous dependency analysis is divided into two approaches. One approach is based on a statistical model [1] [2] [3]. These models need to calculate the probabilities for all possible dependencies in a sentence to obtain the optimal set of dependency. It is not efficient. The other approach is a cascaded chunking model [4] based on SVM [5]. The method is simple and efficient. It achieves high accuracy. The further way to improve accuracy is to increase the training data. However, the increase of training data will bring a great amount of training cost. And the parsing efficiency will be affected as the number of support vectors increased.

In the training data, there are many examples that are unused or not good to improve the classifier's performance. If these examples can be deleted, the training cost will be decreased and the analysis accuracy as well as efficiency will be improved. This paper presents a method of Japanese dependency structure analysis based on improved SVM. First, we train an initial classifier with a small training set, and then prune the large training set with the initial classifier to obtain a small reduction set. And then, we prune the training set continually, reserve or delete a sample according to whether its nearest neighbor has same class label with itself or not. Training with the reduction set, final classifier is obtained. Furthermore, we combine improved SVM with KNN to improve the performance of dependency analyzer. Experiments using the Kyoto University Corpus show that the presented method outperforms previous systems as well as improves the parsing and training efficiency.

2 Cascaded Chunking Model Using SVM

2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) [5] is one of the binary linear classifiers introduced by Vipnik. Suppose l training examples $(\mathbf{x}_i, y_i)$, $(1 \le i \le l)$ are given, where x_i is a feature vector in n dimensional

feature space, y_i is the class label {+1, -1} (positive or negative) of x_i. SVM finds a hyperplane $(\mathbf{w}\cdot\mathbf{x}+b)=0$ which separate the training examples and has maximum margin between two hyperplane $(\mathbf{w}\cdot\mathbf{x}+b)\geq 1$ and $(\mathbf{w}\cdot\mathbf{x}+b)\leq -1$. The optimal hyperplane with maximum margin can be found by solving the following quadratic programming problem.

$$\min \quad \frac{1}{2}\|\mathrm{w}\|^2 \qquad (1)$$

$$\text{subject to} \quad \mathrm{y}_i(w\cdot x_i+b)\geq 0, \quad i=1,2,...,l$$

The decision function can be written as:

$$f(x)=\mathrm{sgn}[\sum_{\mathbf{x}_i\in sv}\alpha_i y_i(x_i\cdot x)+b] \qquad (2)$$

Where α_i is the Lagrange multiplier corresponding to each constraint. The Kernel function $K(x_i,x_j)=\phi(x_i)\cdot\phi(x_j)$ can reduce the computational overhead when the training example x is projected onto a high dimensional space by using projection function ϕ . Among the many kinds of Kernel functions, the *d-th* polynomial kernel: $K(x_i,y_j)=(x_i\cdot x_j+1)^d$ is used. Where d is the dimension of the polynomial functions.

Further more, the optimization problem can be written into the following maximum problem.

$$L(\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j K(x_i\cdot x_j) \qquad (3)$$

Finally, the label of an unknown example is decided by the following function:

$$f(x)=\mathrm{sgn}[\sum_{\mathbf{x}_i\in sv}\alpha_i y_i K(x_i\cdot x)+b] \qquad (4)$$

SVM estimate the label of an unknown example whether sign of $f(x)$ is positive(+1) or negative(-1).

2.2 Japanese dependency Analysis Model

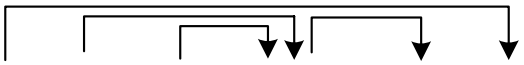
We define a sentence as a sequence of segments $B=\langle b_1,b_2,...,b_m\rangle$ and its syntactic structures as a sequence of dependency patterns $D=\langle dep(1),dep(2),...,dep(m-1)\rangle$, where $dep(i)=j$ means that segment b_i depends on (modifies) segment b_j. In this frame-work, we suppose that the dependency sequence satisfies the following constrains.

1. Except for the rightmost one, each segment depends on exactly one of the segments appearing to the right.
2. Dependencies do not cross each other.

In order to use SVM for dependency analysis, we adopt a sample method: We take a pair of segments that are in a dependency relation as a positive data, and a pair of segments that are not in a dependency relation as a negative data.

Japanese dependency analysis algorithm is as follows:

1. Put an O tag on all segments since the dependency relation of each one is undecided.
2. For each segment with an O tag, decide whether it modifies the segment on its immediate right hand side. If so, the O tag is replaced with a D tag.
3. Delete all segments with a D tag that are immediately followed by a segment with an O tag.
4. Terminate the algorithms if a single segment remains, otherwise return to step 2 and repeat.



母は 私に いろいろな 物を 買ってくれました。
（Mother bought many thing for me . ）

母は 私に いろいろな 物を 買ってくれました。
Mother me many thing buy
Initialization
Input: 母は 私に いろいろな 物を 買ってくれました。
Tag: O O O O O

Input: 母は 私に いろいろな 物を 買ってくれました。
Tag: O O D(Del.) D O

Input: 母は 私に 物を 買ってくれました。
Tag: O D(Del.) D O

Input: 母は 物を 買ってくれました。
Tag: O D(Del.) O

Input: 母は 買ってくれました。
Tag: D(Del.) O

Input: 買ってくれました。
O(Finish)

Fig. 1. Example of the parsing process with cascaded chunking model

Figure 1 shows an example of the parsing process and the result. In training, the model simulated the parsing algorithm by consulting the correct answer from the training annotated corpus. In testing, the model consults the trained system and parses the input sentence with the parsing algorithm.

The simplest and most effective way to improve accuracy is to increase the training data. However, the increase of training data will bring a great amount of

training cost. And the support vectors of the trained model will increase. This will make the analysis time too long to apply the trained model actually.

In the training data, there are many examples that are unused or not good to improve the classifier's performance. If these examples can be deleted, the training cost will be decreased and the analysis accuracy as well as efficiency will be improved.

3 Japanese Dependency Analysis Based on improved SVM

Several attempts have been made to improve classification accuracy of SVM by prune the training set [6] [7]. We adopt the approach called Large training set SVM (LSVM). It prunes the large training set with the initial classifier according to the distance from a sample to the separate hyperplane. And then, we adopt the other approach called Nearest Neighbor-SVM (NN-SVM). It deletes a sample if its nearest neighbor has different class label with itself. In this paper, we use LSVM and NN-SVM (here after we call it NN-LSVM) to prune the training set. Then train with the reduction set to obtain final classifier.

To improve the performance of dependency analyzer further, we combine improved SVM with KNN. In the class phase, the algorithm computes the distance from the test sample to the optimal hyperplane of SVM in feature space. If the distance is greater than the given threshold, the test sample would be classified on SVM; otherwise, the KNN algorithm will be used.

3.1 LSVM

LSVM is a learning strategy of SVM used to large training set. When we use a large training set, in order to reduce the computation cost and do not affect the accuracy of the classifier, we prune the training data as follows:

1. Abstract a small training set S from a large training set L. Train an initial classifier with the small training set S. The size of the small training set S is decided under the following condition:
   (1) The training cost is not big when training with it.
   (2) The accuracy of the classifier is adequacy when training with it.
2. Prune the large training set L with the initial classifier to obtain a small reduction set. Then training with the reduction set, finial classifier is obtained. The prune way is shown in figure 2.

The separate hyperplane of the initial classifier is H. The distance from a sample S to the separate hyperplane H is $d(d \geq 0)$. Remain the sample if $1-\varepsilon < d < 1+\varepsilon$. Otherwise, delete it. We can control the size of the reduce set and the classifier accuracy by adjust the threshold $\varepsilon(0 < \varepsilon < 1)$. In fact, the optimal classifier can be obtained by adjust the threshold ε .

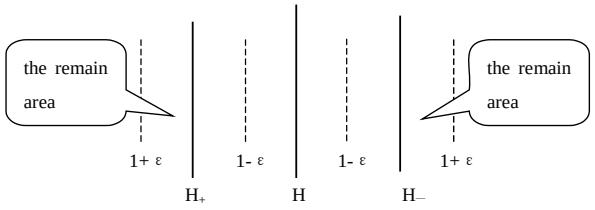


Fig. 2. The pruning figure of the LSVM

This prune strategy is designed along with that the classifier is related with the support vectors only and unrelated with the other samples. After pruning with this strategy, the remained samples are useful to classification and the deleted samples are useless or even reactive to classification.

3.2 NN-SVM

SVM focuses on the samples near the boundary in training time, and those samples intermixed in another class are usually no good to improve the performance of classier. On the contrary, they may greatly increase the cost of computation and their existence may lead to overlearning and decrease the generalization ability.

NN-SVM is an improved SVM that can improve the generalization ability of classifier. It reserves or deletes a sample according to whether its nearest neighbor has same class label with itself or not, then trains the new set with SVM to obtain a classifier.

We prune the training data as follows:

Find the nearest neighbor of each sample. If a sample has the same class with its nearest neighbor reserve it, otherwise delete it.

The distance between the two vectors $x_i(x_i = (x_i^1, x_i^2, \cdots, x_i^n))$ and $x_j(x_j = (x_j^1, x_j^2, \cdots, x_j^n))$ is as:

$$D(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_i^k - x_j^k)^2} \qquad (5)$$

3.3 SVM-KNN

Analyzing the classification results, we find that the misclassified samples are mainly near the hyperplane. This means that we should use the information of the samples near the optimal hyperplane to improve the performance of the classifier. Based on SVM theory, the samples near the optimal hyperplane are mainly support vectors. In the meantime, the classifier based on SVM is regard as a 1NN classifier in which only one representative point is selected for each class. Therefore, A SVM-KNN classifier combined SVM with KNN is presented to improve the performance of SVM classifier [8]. The algorithm of SVM-KNN is described as follows:

Suppose T is the testing set.

(1) if $T \neq \Phi$, select $x \in T$, else stop;

(2) compute the distance $g(x)$ from the test sample to the hyperplane of SVM as described in (6);

(3) if $g(x) > \varepsilon$, $\varepsilon \in [0,1]$, output $f(x) = sgn(g(x))$, else use KNN algorithms and output the returned results;

(4) $T \leftarrow T - \{x\}$, go to step (1)

The distance from the test sample to the hyperplane of SVM in feature space is as:

$$g(x) = \sum_{x_i \in sv} y_i \alpha_i K(x_i, x) - b \quad (6)$$

The distance from each test sample to each reference point is as:

$$d(x, x_i) = \left\| \phi(x) - \phi(x_i) \right\| = \sqrt{K(x,x) - 2K(x,x_i) + K(x_i,x_i)} \quad (7)$$

Where x_i is the support vector.

4 Experiments And Discussion

4.1 Experiments Setting

We use Kyoto University text corpus (Version 3.0) consisting of articles of Mainichi Newspaper. The sentences from the articles on January 1st, 3rd to 9th are used for the training data, and the sentences from the articles on January 10th are used for the test data. Our experiments are under the condition $d = 3$ (dimension of the polynomial functions used for the Kernel function).

The features used in the dependency parsing process are shown in Table 1. The features include static features and the dynamic features

4.2 Experimental Results

4.2.1 Based on improved SVM

We train an initial classifier with the sentences from articles on January 1st. Then we prune the data on January 1st, 3rd to 9th with LSVM ($\varepsilon = 0.9$, experiments show that the parsing accuracy is best when $\varepsilon = 0.9$). We prune the training data continue with NN-SVM and obtained the final classifier. Test the data on January 10th based on SVM, LSVM and NN-LSVM. The experimental results are shown in table 2.

Table 2 shows the dependency accuracy and the sentence accuracy are improved using large training data set based on SVM(86.86%→89.29%, 41.50%→ 47.53%). The time required for training and parsing are significantly increased(4minutes → 908minutes, 0.26sec./sentence→1.7sec./sentence).

Table 1. Features used in the dependency parsing process

Static Features	left/right segments	Head Word (surface-form, POS, POS-subcategory, inflection-type, inflection-form), Functional Word (surface-form, POS, POS-subcategory, inflection-type, inflection-form), brackets, quotation-marks, punctuation-marks, position in sentence (beginning, end)
	Between two segments	Distance (1,2-5,6-), case-particles, brackets, quotation-marks, punctuation-marks
Dynamic Features	The segments which modify the current candidate modifee or modifer	Form of inflection represented with Functional Representation
	The segment which is modified by the current candidate modifee	POS and POS-subcategory of Head word

We use LSVM and NN-LSVM to prune the training data. Even though the number of examples used for LSVM and NN-LSVM is less than that for SVM, dependency accuracy and sentence accuracy are improved(89.296% → 89.86%, 47.53% → 49.14%). And the time required for training and parsing are evidently reduced((908minutes → 129minutes, 1.7sec./sentence→0.8sec./sentence).

Experiments show that the improved SVM not only reduces the training cost greatly but also obtains a classifier that has better accuracy and efficiency than the classifier obtained by training large set directly.

Training with LSVM increases the training cost of adjusting the threshold. However the increased cost is time cost instead of memory cost. This cost increases linearly. SVM requires O(n2) training cost, (where n is the number of examples.). It includes time cost and memory cost, and increases with the square of the number of examples.

Table 2. Results based on NN-LSVM

Model Training data (days)	SVM 1	SVM 8	LSVM 8	NN-LSVM 8
Dependency Acc. (%)	86.86	89.29	89.35	89.86
Sentence Acc. (%)	41.50	47.53	47.58	49.14
Parsing Time (sec./ sentence)	0.26	1.7	1.5	0.8
The number of support vectors	5485	35181	32306	17165
The number of examples	15052	132022	104380	97727
Training Time (minutes)	4	908	643	129

4.2.2 Combined improved SVM with KNN

Table 3 shows that the approach combined improve SVM and KNN achieved higher dependency accuracy and sentence accuracy than the improved SVM when k is between 27 and 33 in the confidence $\varepsilon = 0.1$. However, the approach combined improved SVM and KNN cannot performance better than the improved SVM when $\varepsilon \geq 0.2$. This means that the misclassified samples are mainly near the hyperplane.

Table 3. Results based on improved SVM and KNN

	Dep. Acc. (%) (ε =0.1)	Sen. Acc. (%) (ε =0.1)	Dep. Acc. (%) (ε =0.2)	Sen. Acc. (%) (ε =0.2)
K=1	89.75	48.29	89.69	48.02
K=5	89.70	48.35	89.68	48.15
K=10	89.75	48.96	89.65	48.96
K=20	89.78	48.22	89.41	47.41
K=25	89.91	48.82	89.72	48.49
K=26	89.98	49.09	89.81	48.62
K=27	90.02	49.23	89.84	48.69
K=28	90.01	49.16	89.84	48.96
K=29	90.00	49.16	89.84	48.76
K=33	90.00	49.16	89.84	48.76

4.3 Comparison with Related Work

The results of our model and the recent Japanese Dependency Analysis model (cascaded chunking [3], ME [1], ME + posterior context [2]) are summarized in table 4. Dependency accuracy and sentence accuracy are improved using improved SVM and KNN. The time required for training and parsing cannot be compared since the computers used for each experiment are different.

Table 4. Comparison with the related work

Model	Training Corpus (# of days)	Dependency Acc. (%)	Sentence Acc. (%)
Improved SVM and KNN	Kyoto Univ. (8)	90.02	49.23
Improved SVM	Kyoto Univ. (8)	89.86	49.14
Cascaded chunking[4]	Kyoto Univ. (8)	89.29	47.53
Probabilistic (ME)[1]	Kyoto Univ. (8)	87.14	40.60
Probabilistic (ME + posterior context)[2]	Kyoto Univ. (8)	87.93	43.58

6 Conclusion

This paper presented a method for Japanese dependency analysis that using improved SVM and KNN. The improved SVM delete those samples that are unused or not good to improve the classifier's performance, and then trained with the reduction set to obtain the final classifier. Experiments show that the improved SVM not only reduces the training cost

greatly but also obtains a Japanese analyzer that outperforms the analyzer obtained by training large set directly with respect to accuracy and efficiency. Furthermore, we combined improved SVM and KNN. The experimental results show that that the mixed algorithm can improve the dependency accuracy compared to improved SVM.

References:

[1] Kiyotaka Uchimoto, Satoshi Sekine, Hitoshi Isahara, Japanese Dependency Structure Analysis Based on Maximum Entropy Models. In Proceedings of the EACL, 1999, pp. 196-203

[2] Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, Hitoshi Isahara, Dependency Model using posterior context. In Proceedings of Sixth International Workshop on Parsing Technologies, 2000, pp. 321-322

[3] Taku Kuto, Yuji Matsumoto, Japanese Dependency Structure Analysis Based on Support Vector Machines. In Empirical Methods in Natural Language processing and Very Large Corpora, 2000, pp. 18-25

[4] Taku Kuto, Yuji Matsumoto, Japanese Dependency Analysis using Cascaded Chunking. In Conference on Computational Natural Language Learning, Taipei, Taiwan, 2002, pp. 63-69

[5] Vapnik, V.N., The Nature of Statistical Learning Theory. Springer-Verlag, Berlin, 1995

[6] LI Hong-Lian, WANG Chun-Hua, YUAN Bao_Zong, ZHU Zhan-Hui, A Learning Strategy of SVM to Used Large Training Set. Chinese Journal of Computer, Vol. 27, 2004, pp.715-719

[7] LI Hong-Lian, WANG Chun-Hua, YUAN Bao-Zong: An Improved SVM, NN-SVM. Chinese Journal of Computer, Vol. 26, 2003, pp.1015-1020

[8] LI Rong, YE Shi-wei, and SHI Zhong-zhi, SVM-KNN Classifier-a New Method of Improving the Accurracy of SVM Classifier, Acta Electronica Sinica, vol. 30, 2002, pp.745-758