Dependency Analysis of Japanese Spoken Language via SVM

Kiyonori Ohtake
ATR Spoken Language Translation Research Laboratories
2-2-2 Hikaridai, Keihanna Science City, Kyoto 619-0288 Japan
E-mail: kiyonori.ohtake@atr.co.jp

Abstract

This paper discuss a dependency analyzer employing Support Vector Machines (SVMs) for Japanese spoken language. Most conventional dependency analyzers target written texts. Thus, we use a currently available spoken language corpus and make the SVMs learn the corpus to build a dependency analyzer that targets spoken language. We used two types of corpora: one contains written language, and the other, spoken language. By repeating closed testing procedures via the SVMs, we cleaned the corpora. In addition, the cleaning raised the accuracy of the dependency structure analysis by two-fold cross validation over 1% for written language, and 0.25% for spoken language.

Keywords: dependency analysis, spoken language corpus, support vector machines, corpus-based method.

1 Introduction

Dependency analysis is one of the most important techniques for natural language processing. Thus, so far, many works have been done on this subject, and several analyzers have been developed. However, most of these analyzers target written texts. There are few analyzers that target spoken language (e.g., (Matsubara et al., 2002)), in part because in contrast to written texts there are not many spoken language corpora. In addition, it makes analyzing grammar difficult because there are several phenomena in spoken language that are not seen in written texts.

Some recent works encourage corpus-based dependency analyzing (e.g.,(Uchimoto et al., 1999; Kudo and Matsumoto, 2002; Kanayama et al., 1999)). If we prepare a correct dependency-analyzed corpus, we can build a dependency analyzer. However, there are few works that discuss corpus-based dependency analysis for spoken Japanese. Consequently, we use the ATR Speech and Language DataBase[1] (SLDB) as a currently available dependency analyzed corpus of Japanese spoken language, and attempt to build a dependency analyzer for spoken Japanese via SVMs. In this paper, we determine the accuracy of a corpus-based dependency analyzer for spoken Japanese via SVM with a currently available spoken corpus. We also employ a rule-based dependency analyzer, KNP[2], to compare the two approaches: rule-based and SVM.

In addition, we discuss corpus cleaning via SVM because currently available corpora contain errors that cause problems. We prepared a tagging tool for our dependency analyzer to correct the dependency-analyzed corpus.

2 Dependency Analyzers

We use two dependency analyzers. One is a rule-based analyzer, and the other employs SVMs.

[1] http://www.red.atr.co.jp/database_page/taiwa.html
[2] http://www.kc.t.u-tokyo.ac.jp/nl-resource/knp.html

The Japanese dependency structure is usually defined in terms of the relationship between phrasal units called *bunsetu* segments. Conventional methods of dependency analysis have assumed the following three syntactic constraints (Kurohashi and Nagao, 1994a):

1. All dependencies are directed from left to right.
2. Dependencies do not cross each other.
3. Each *bunsetu* segment, except the last one, depends on only one *bunsetu* segment.

Matsubara et al. claim that an investigation into a spoken dialogue corpus proved that many spoken utterances do not satisfy these constraints (Matsubara et al., 2002). Therefore, they proposed a dependency analyzing method that relaxes the first and third of the above three constraints.

However, we believe that there is no need to relax these constraints to analyze spoken language because most situations in which we have to consider the direction of a dependency from right to left may be considered as inversions. We can deal with those situations with two approaches. One is to detect inversions and process them individually. The other is to assume that a proper unit does not need to consider the inverse direction of dependency, then consider the relationship between the units. In other words, this unit corresponds to a unit currently called a sentence, and it is natural that we view the phenomenon as the relationship between sentences because the order in the utterance expresses some meaning.

In addition, in the investigation by Matsubara et al., most of the *bunsetu* segments that depend on no other segments are considered fillers. Therefore there is no need to relax the third constraint because we can individually pre-process these fillers.

2.1 CaboCha: dependency analyzer via SVM

CaboCha[3] is a dependency analyzer that employs SVMs with a cascaded chunking model. The analyzer was developed by Kudo and Matsumoto(Kudo and Matsumoto, 2002).

One characteristic of the SVMs is their high generalization ability. Thus, SVMs are not only included in the field of natural language processing, but also in many other fields wherein machine learning techniques are effective, such as pattern recognition.

The overview of dependency analysis by CaboCha is described as follows:

1. Morphological analysis is applied to the input sentence by using Chasen[4].
2. Chunk the morphemes into a series of *bunsetu* segments by an SVM-based chunker[5].
3. Analyze the relationship between the *bunsetu* segments with an SVM.

The analyzing method for the relationship between *bunsetu* segments by a cascaded chunking model is as follows:

1. Put an **O** tag on all segments. The **O** tag indicates that the dependency relation of the current segment is undecided.
2. For each segment with an **O** tag, decide by an SVM 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 algorithm if a single segment remains, otherwise return to step 2 and repeat.

[3] `http://cl.aist-nara.ac.jp/~taku-ku/software/cabocha/`
[4] `http://chasen.aist-nara.ac.jp/`
[5] The chunker is YamCha, available at `http://cl.aist-nara.ac.jp/~taku-ku/software/yamcha/`.

The cascaded chunking model can be combined with any machine learning algorithm that works as a binary classifier, since the cascaded chunking model parses a sentence deterministically only decideing whether or not the current segment modifies the segment on its immediate right hand side. CaboCha uses SVMs as a machine learning algorithm because of their state-of-the-art perfomance and generalization ability.

CaboCha uses two kinds of features for machine learning. One is referred to as static features: head words and their parts-of-speech, functional words and inflection forms of the words that appear at the end of segments, the distance between two segments, and the existence of punctuation marks. The other is called dynamic features that are created dynamically during the parsing process. The following three types of features are used in CaboCha: (A) The segments that modify the current candidate modifiee, (B) the segements that modify the current candidate modifier, and (C) the segment that is modified by the current candidate modifiee.

CaboCha, learned from the Kyoto University Text Corpus (hereafter KUTC) performs with almost a 90% accuracy by two-fold cross validation.

2.2 KNP: rule-based dependency analyzer

KNP (Kurohashi Nagao Parser) is a rule-based Japanese dependency analyzer developed by Kurohashi and Nagao(Kurohashi and Nagao, 1994b). The KNP analysis method first detects conjunctive structures in a sentence by checking the parallelism of two series of words and then analyzes the dependency structure of the sentence with the help of information about the conjunctive structures.

An overview of dependency analysis by KNP can be described as follows:

1. Morphological analysis is applied to the input sentence by using JUMAN[6].
2. Chunk the morphemes into a series of *bunsetu* segments by manually built rules.
3. Detect each scope of the conjunctive structures of the input sentence by detecting a distinctive key *bunsetu* that accompanies one of the pre-determined expressions or has a continuous form, and comparing similarities between two series of *bunsetus*.
4. Analyze the dependency structures in each conjunctive structure by determining the head *bunsetu* from right to left for each *bunsetu* based on several heuristics.

There is no published report that rigidly evaluates the performance of KNP and compares it with other analyzers for written language. However, we empirically know that the accuracy of dependency analysis by KNP is almost 90% for newspaper articles.

3 Corpora

We employ two tagged corpora, KUTC (Kyoto University Text Corpus) and ATR SLDB. KUTC is a corpus of newspapers. Newspaper articles were automatically analyzed, and were checked and corrected manually. On the other hand, ATR SLDB contains formal travel-type conversations between two persons. The conversations were manually transcribed and automatically analyzed, and were also checked and corrected manually.

KUTC (version 3.0) contains almost 40,000 sentences consisting of *Mainichi Shimbun* (Mainichi Newspaper) articles from January 1st to January 17th 1995 and all editorials of 1995. The articles were automatically analyzed by the JUMAN morphological analyzer and the KNP dependency analyzer. After the automatic analysis, the analyzed results were checked and corrected by several people.

[6] `http://www.kc.t.u-tokyo.ac.jp/nl-resource/juman.html`

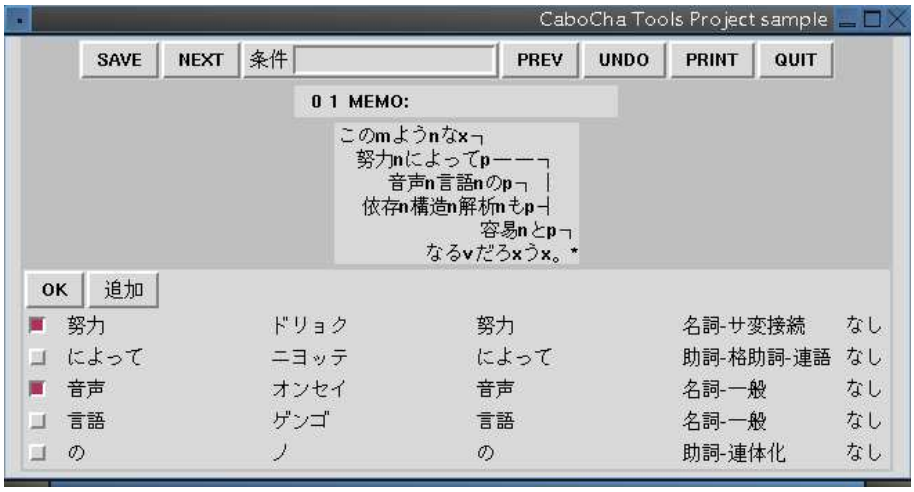

Figure 1: Corpus tagging tool interface

The ATR SLDB contains 618 conversations including 21,761 utterances. We used JDEP in the ATR SLDB, which contains dependency analyzed results. Unlike most Japanese dependency analysis situations, JDEP's dependency structure unit is not a *bunsetu* segment, but a morpheme. Thus, we first analyzed JDEP with CaboCha (normal setting), and aligned the dependency structures of the results with JDEP. Finally, we manually and roughly checked the *bunsetu* segments in all utterances. We roughly checked the results because we know the corpus cleaning procedure detects bad segmentations.

4 Cleaning Corpora

Both corpora include some analysis errors, or fluctuations. The errors and fluctuations cause a decrease in the analyzer's accuracy if we employ a statistical corpus-based analyzer.

Our cleaning method is very simple. We just execute closed testing procedures with CaboCha and manually correct the errors found. Closed testing procedures find some careless errors and fluctuations, and there are two levels of closed testing procedures. One is the surface chunking level (*bunsetu* segmentation), and the other is the dependency structure level. To correct an error in a corpus, we prepared a corpus tagging tool for CaboCha which is similar to that of KUTC. An example of the tool's interface is shown in Figure 1.

To correct the corpus, we followed the standards of the KUTC guidelines for corpus building. However, when we cleaned JDEP, there were many phenomena, especially seen in spoken language, that were not covered by the guidelines. In those cases, we individually judged and modified the standards.

4.1 Cleaning KUTC

The POS system that is used to analyze KUTC is the JUMAN system. On the other hand, the POS system of CaboCha is not the JUMAN system because the normal setting of CaboCha utilizes Chasen as a morphological analyzer. These POS systems are slightly different. Meanwhile, CaboCha is also able to use the JUMAN system as a POS system. However, we empirically know that the Chasen's POS system is more suited to spoken language than that of JUMAN. Thus, we built a POS system converter from the JUMAN system to the Chasen system, and prepared KUTC with the Chasen system.

First, we executed closed testing procedures for *bunsetu* segments. We repeated the closed tests and manual corrections until the accuracy of the result reached 100%. Second, we executed a closed testing procedure for dependency structures just once, and manually corrected the errors detected by the test.

We conducted two-fold cross validations for dependency analysis to investigate the effects of corpus cleaning. We divided the corpus in half using the date of each article. As a result, there were some sentences included in both learning data and test data because some articles were shared by both national editions and local editions, and the corpus includes both editions. However, such sentences were very rare relative to other sentences. The cleaning results for KUTC are shown in Table 1. In Table 1, the accuracy means the dependency accuracy corresponding to the percentage of correct dependencies out of all dependencies.

Table 1: Cleaning effects for dependency analysis

	Accuracy (acc.)	
	KUTC	JDEP
Before cleaning	88.85%	92.78%
After cleaning	89.92%	93.03%

4.2 Cleaning ATR SLDB

The original POS system of JDEP (ATR SLDB) is different from that of CaboCha. However, we had already analyzed JDEP by CaboCha in order to manage the difference in units for dependency analysis. Consequently, there was no need to consider the difference in the POS systems.

As with the cleaning of KUTC, first, we executed closed testing procedures for *bunsetu* segments and repeated closed tests and manual corrections until the accuracy of the result reached 100%. Second, we executed a closed testing procedure for the dependency structure just once, and manually corrected the detected errors.

We also conducted two-fold cross validations for dependency analysis to investigate the effects of corpus cleaning. Unlike the situation for KUTC, we eliminated the duplicated sentences in JDEP, and conducted two-fold cross validations. Thus, the experimental condition for JDEP is stricter than that for KUTC. The JDEP cleaning results are shown in Table 1.

Table 1 shows that the effect of cleaning on JDEP is slightly smaller than that on KUTC. However, there were several improvements that were not shown in the accuracy of dependency analysis. For example, there were many errors in *bunsetu* segmentation before the cleaning. After cleaning, most of them were corrected.

5 Experiments

We carried out two experiments. One was performed using the default settings for both dependency analyzers, CaboCha and KNP. The other was performed by using cleaned corpora with CaboCha.

First, we analyzed KUTC and JDEP by both CaboCha and KNP under the normal settings. The normal setting for CaboCha means that the learning corpus for *bunsetu* segment chunking and dependency analysis is KUTC (Chasen POS system) prepared by Kudo et al. Table 2 shows the results.

CaboCha's result for KUTC in Table 2 indicates a closed test result. However, the original corpus, which was used to build CaboCha's normal model, and KUTC, which we had prepared for this study, were slightly different. Thus the CaboCha's result for KUTC shown in Table 2 may be lower than the result of a proper closed test. In Tables 2 and 3, the sentence accuracy means the percentage of sentences in which all dependencies were analyzed correctly.

Table 2: Analyzed results for KUTC and JDEP with the normal settings

	KUTC		JDEP	
	Dependency acc.	Sentence acc.	Dependency acc.	Sentence acc.
KNP	91.23%	57.17%	84.27%	69.54%
CaboCha	97.36%	83.40%	88.43%	76.01%

Second, the CaboCha that learned with the cleaned corpora analyzed both learning corpora. Table 3 shows the results.

Table 3: Analyzed results by CaboCha with cleaned corpora for KUTC and JDEP

Corpus	Dependency acc.	Sentence acc.
KUTC	99.86%	99.08%
JDEP	99.56%	98.94%

6 Discussions

The cleaning corpora via closed testing procedures with SVMs raised the dependency analysis accuracy over 1% for KUTC and 0.2% for JDEP. The average length per sentence of JDEP is shorter than that of KUTC (JDEP: 11.8 morphemes/sentence, KUTC: 26.6 morphemes/sentence). Consequently, building the original dependency structure for JDEP was easier than that for KUTC, and there were fewer mis-analyses.

We had assumed that the results of analyses for JDEP by both KNP and CaboCha with the normal settings would be very low because the analyzers were developed for written language rather than spoken language. However, the results shown in Table 2 are not very valid because the average length per sentence of JDEP is shorter than that of KUTC, and it is easy to attain the accuracy of the dependency structure.

We can infer from the results shown in Table 2 that the SVM is highly capable at generalizing, because despite the fact that CaboCha learned the KUTC corpus, namely written texts, the results showed higher values than we expected.

Table 3 shows that CaboCha with cleaned corpora performed with very high accuracy in its closed testing procedures, and one may think that it was overfitted because the closed testing procedure results reached almost 100%. However, we did not change any parameters and did not employ any other models; we just corrected errors.

Meanwhile, there were several differences between KUTC and JDEP. We arranged some *bunsetu* segmentations to analyze JDEP. For example, "*go touchaku ni naru*" is the honorific form of "*tsuku*" (to arrive), and CaboCha learned with KUTC segments the expression "*go tochaku ni /naru*." However, this segmentation causes some problems. For instance, "*hotel ni go touchaku ni naru*" (arrived at a hotel) is segmented into "*hotel ni /go touchaku ni /naru*," and it is ambiguous whether "*hotel ni*" (at a hotel) depends on "*go touchaku ni*" or "*naru*."

It is natural that we view "*go touchaku ni naru*" as one *bunsetu* segment because "*go touchaku ni naru*" plays the role of the verb "*tsuku*." Thus, we connected several *bunsetu* segments that played one functional role as one segment in JDEP. As a result of this arrangement, CaboCha needs longer pieces of information (morpheme sequences) than before to judge a segment.

7 Concluding Remarks

This paper discussed a dependency analyzer employing SVMs for spoken Japanese. We used two types of corpora. One was a written-language corpus, KUTC (Kyoto University Text Corpus) and the other a spoken-language corpus, JDEP in the ATR SLDB. Repeated closed testing procedures via SVM were used to clean the corpora, and finally the results showed that the accuracies of dependency-structure analysis reached over 99% for both corpora. In addition, cleaning raised the accuracy of dependency structure analysis by two-fold cross validation.

The accuracy of dependency structure analysis by two-fold cross validation for JDEP was over 93%. The value was higher than that for KUTC because the average length per sentence of JDEP is shorter than that of KUTC, thereby making it easy to achieve high accuracy.

JDEP, the dependency analyzed spoken corpus that we used, is very clean. There are few grammatically ill-formed linguistic phenomena such as fillers, hesitations and self-repairs. Most of them were rejected when the corpus was transcribed from speech to build the dependency structures. However, to analyze spontaneous Japanese spoken language in the real world, we have to detect such phenomena and know how to handle them. Coping with these phenomena in order to analyze spontaneous Japanese spoken language is left as a future topic for study.

Acknowledgement

This research was supported in part by the Telecommunications Advancement Organization of Japan.

References

Hiroshi Kanayama, Kentaro Torisawa, Yutaka Mitsuishi, and Jun'ichi Tsujii. 1999. Statistical dependency analysis with an HPSG-based Japanese grammar. In *Proceedings of Natural Language Processing Pacific-Rim Symposium (NLPRS '99)*, pages 138–143.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *CoNLL 2002: Proceedings of the 6th Conference on Natural Language Learning 2002 (COLING 2002 Post-Conference Workshops)*, pages 63–69.

Sadao Kurohashi and Makoto Nagao. 1994a. KN Parser: Japanese dependency/case structure analyzer. In *Proceedings of Workshop on Sharable Natural Language Resources*, pages 48–55.

Sadao Kurohashi and Makoto Nagao. 1994b. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Shigeki Matsubara, Takahisa Murase, Nobuo Kawaguchi, and Yasuyoshi Inagaki. 2002. Stochastic dependency parsing of spontaneous Japanese spoken language. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-2002)*, pages 640–645.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proceedings of 9th European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 196–203.