Phrase-Based Dependency Evaluation of a Japanese Parser

Hisami Suzuki

Microsoft Research
One Microsoft Way, Redmond WA 98052 USA
hisamis@microsoft.com

Abstract

Extraction of predicate-argument structure is an important task that requires evaluation for many applications, yet annotated resources of predicate-argument structure are currently scarce, especially for languages other than English. This paper presents an evaluation of a Japanese parser based on dependency relations as proposed by Lin (1995, 1998), but using phrase dependency instead of word dependency. Phrase-based dependency analysis has been the preferred form of Japanese syntactic analysis, yet the use of annotated resources in this format has so far been limited to training and evaluation of dependency analyzers. We will show that (1) evaluation based on phrase-dependency is particularly well-suited for Japanese, even for an evaluation of phrase-structure grammar, and that (2) in spite of shortcomings, the proposed evaluation method has the advantage of utilizing currently available surface-based annotations in a way that is relevant to predicate-argument structure.

Introduction

Predicate-argument structure is a linguistically significant structure that could potentially benefit many linguistically motivated applications. Therefore, evaluating the predicate-argument structure as an output of a system is an important task, both for measuring system-internal improvements over time, and for conducting a cross-system evaluation. However, gold standard annotation for predicate-argument structure is difficult to come by, especially for non-European languages. In this paper, we present a method of evaluation that expands on the idea of dependency-based evaluation proposed by Lin (1995, 1998); but instead of using word dependency as Lin did, we use phrase dependency. We will show that phrase-based dependency evaluation works particularly well in Japanese even for an evaluation of a phrase-structure grammar, and that it has the advantage of utilizing currently available surface-based annotations in a way that is directly relevant to predicate-argument structure.

Dependency-Based Evaluation

Word-based dependency

Lin (1995) proposes a dependency-based evaluation metric for English parsers, which measures the dependency between two words in a sentence. For example, for the sentence "I saw a bird with a telescope", the following word::headword pairs are extracted for the correct analysis (1a) and for an analysis with an attachment error (1b), that is, the prepositional phrase *with a telescope* is attached wrongly to *bird*:[1]

(1a) (I :: saw) (saw :: *) (a :: bird) (bird :: saw) (**with :: saw**) (a :: telescope) (telescope :: with)

(1b) (I :: saw) (saw :: *) (a :: bird) (bird :: saw) (**with :: bird**) (a :: telescope) (telescope :: with)

Precision and recall can be computed based on the correct dependency pairs and the pairs produced by a given system. In the case above, both precision and recall are 6/7 ≈ 85.7%.[2] Among the desirable properties of dependency-based evaluation that Lin (1995) describes, the most relevant for the identification of predicate-argument structure is that the dependency-based evaluation reflects the appropriateness of a parse at the predicate-argument structure level more truthfully. Consider the same example as in (1) in the evaluation framework based on phrase boundary. (2a) is the correct phrase bracketing, while (2b) is the bracketing for the analysis with the same PP-attachment error:

(2a) [I [saw **[**a **[**bird**]]** [with [a [telescope]]]]]

(2b) [I [saw **[**a **[**bird [with [a [telescope]]]**]]**]]

The accuracy of this analysis (both recall and precision) according to the bracketing-based metric is 5/7 ≈ 71.4%. It is lower than the accuracy figure computed by the dependency-based metric, because the attachment mistake, which counts as a single mistake in a predicate-argument structure, is counted twice. The problem of counting an attachment mistake multiple times only becomes worse as there are more modifiers to the word to which the prepositional phrase is wrongly attached.[3] This is the major reason why word-dependency-based metric reflects semantically meaningful dependencies more faithfully than the metric based on constituent bracketing.

Phrase-based dependency

However, dependency-based annotation based on words cannot be easily adopted for evaluating parsers for languages like Japanese, because there is an issue of word segmentation: what counts as a word varies significantly depending upon the specification of an annotation and the lexicon of a system. Therefore, we have implemented an evaluation method based on phrase dependency rather than word dependency. Here, the notion of phrase corresponds to the traditional notion of *bunsetsu* in Japanese, which is defined as one content word (or *n*-

[1] * means that the word on the left-hand side is the sentential head.

[2] Precision and recall will be the same as long as there is no ambiguity in tokenization.

[3] The bracketing given in (2) deviates from the Penn Treebank bracketing convention in that the bracketing is indicated within an NP; however, the problem of counting an attachment error multiple times still holds, albeit to a smaller degree, with the Penn Treebank annotation as well.

content words in the case of compounds with *n*-components) plus any number of function words (including postpositions, auxiliaries and affixes). A bunsetsu has the property of being a prosodic unit, in that each bunsetsu can have only up to one accent. The existence of this prosodic property greatly contributes to having consistent identification of bunsetsu as a unit inter-subjectively and across different computational systems, as we will see below.

Given the definition of bunsetsu above,[4] a sentence can be segmented into a sequence of non-overlapping bunsetsu. (3) below presents an equivalent English phrase dependency for the same sentence used in (1): (3a) is for the correct analysis, and (3b) for a system output with a PP-attachment error, with parentheses indicating phrases:

(3a) (I :: saw) (saw :: *) (a bird :: saw) (**with a telescope :: saw**)

(3b) (I :: saw) (saw :: *) (a bird :: saw) (**with a telescope :: bird**)

Using phrase-based dependency for parser evaluation has a number of advantages, especially for evaluating Japanese. Most importantly, it is independent of phrase-internal word- or morpheme-breaking specification, so an annotation can be used for evaluating systems with different underlying grammatical and parsing theories. Another advantage is that phrase-based dependency reflects semantic dependency more directly than word dependency, as it exclusively looks at relationships between content words. For example, a local dependency such as *a::bird* is not evaluated in (3), so a mistake in a semantically meaningful dependency counts more severely in phrase-based dependency than in word-based dependency. The accuracy of the same system output in the above example in word-based metric is 85.7%, while it is 75% (that is, 3/4) in the phrase-based metric in (3).

Needless to say, phrase-based dependency evaluation does not evaluate all dependencies relevant to predicate-argument structure: for example, the proposed method does not take into account the cases where there is more than one predicate or argument within a phrase as in the case of complex predicates, which is quite common in Japanese. The example below is a case in point: the phrase 加入させよ *kanyuu-sase-yo* 'join-CAUS-IMPR' contains two predicates, 加入 *kanyuu* 'join' and させる *saseru* 'let', each of which takes distinct sets of arguments. The phrase-based dependency, as shown in (4b), cannot capture the arguments of the complex predicate properly.

(4a) ＮＡＴＯはこの四カ国を加入させよ。
NATO-wa kono 4kakoku-wo kanyuu-sase-yo
NATO-TOP these 4-countries-ACC join-CAUS-IMPR
'NATO should let these 4 countries join (their organization).'

(4b)	
ＮＡＴＯは::加入させよ	(NATO::join-CAUS-IMPR)
この::四カ国を	(these::4_countries)
四カ国を::加入させよ	(4_countries::join-CAUS-IMPR)
加入させよ::*	(join-CAUS-IMPR::*)

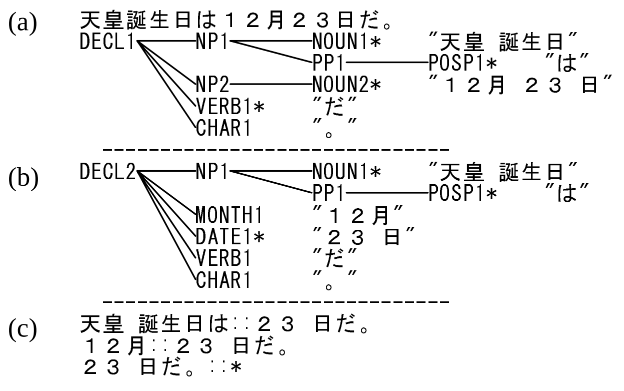
[4] The definition of bunsetsu given here is very similar to the notion of *chunk* proposed by Abney (1991), and is practically identical to that of *ϕ-phrase* by Gee and Grosjean (1983), which Abney's notion of chunk is based on.

However, we believe that the phrase-dependency evaluation is a reasonable first step: although a correct phrase dependency does not guarantee a correct predicate-argument structure, mistakes in phrase dependency always suggest problems in the predicate-argument analysis.

Phrase-Dependency-Based Evaluation of NLPWin-Japanese

In this section, we describe the experiment in which we applied the proposed phrase-based evaluation on a phrase-structure grammar of Japanese. NLPWin-Japanese is a parser under development at Microsoft Research; it has multiple levels of analysis as its output, including surface constituent structure, language-neutral syntax and logical form or LF (Heidorn, 2000; Campbell and Suzuki, 2002). LF is the level of representation that can be considered as the predicate-argument representation within our system. Many of the applications we are interested in, including machine translation and automatic summarization, use LF as their input, hence the need for evaluating LF. However, there is no currently available external resource of annotations equivalent to Japanese LF, therefore an alternative evaluation scheme is called for.

The method we adopted is to map the constituent structure to a dependency structure, and compare it with the dependency structure extracted from Kyoto University Text Corpus (version 3.0, henceforth KC; Kurohashi and Nagao, 1997), which is a bunsetsu-dependency annotated corpus of about 38,000 sentences of Mainichi Newspaper articles in 1995. We implemented the mapping in two steps: (1) Modify the NLPWin tree structure to absorb specification differences by a series of rules; and (2) Compute each phrase and its parent phrase based on the tree given by (1). Figure 1 below shows an example: the surface constituent structure (a) is converted into a tree structure that reflects KC specification (b); then, phrases and their parent phrases are read off from the converted tree and printed in phrase::parent_phrase format (c). These dependency pairs are then compared with the pairs extracted from the KC annotation, after removing white spaces between words, special symbols and punctuation marks.



```
(a)  天皇誕生日は１２月２３日だ。
     DECL1───NP1────NOUN1*    ″天皇 誕生日″
                  └─PP1─────POSP1*   ″は″
          ├─NP2────NOUN2*    ″１２月 ２３ 日″
          ├─VERB1*   ″だ″
          └─CHAR1    ″。″
      ------------------------------
(b)  DECL2───NP1────NOUN1*    ″天皇 誕生日″
                  └─PP1─────POSP1*   ″は″
          ├─MONTH1   ″１２月″
          ├─DATE1*   ″２３ 日″
          ├─VERB1    ″だ″
          └─CHAR1    ″。″
      ------------------------------
(c)  天皇 誕生日は::２３ 日だ。
     １２月::２３ 日だ。
     ２３ 日だ。::*
```

Figure 1: Mapping NLPWin output to KC-style dependency pairs

Using this method, we tested NLPWin-Japanese on 600 KC sentence, including 300 daily articles and 300 editorials.[5] Table 1 below summarizes the results.

corpus	# of sentence	precision	recall
daily article	300	76.86%	78.83%
editorial	300	79.51%	80.45%

Table 1: Phrase-dependency based evaluation of NLPWin

Interpreting Phrase-Dependency-Based Evaluation

The numbers presented in Table 1 are about 10% lower than the numbers reported for state-of-the-art bunsetsu dependency analyzers in Japanese, such as KNP (1994) and CaboCha (2002). In order to interpret these numbers in a cross-system comparison setting, however, we must proceed with due care. In this section, we discuss some caveats that are necessary for reliably interpreting the evaluation results.

First of all, KNP, CaboCha and NLPWin all present different degrees of familiarity with KC: KC is created by manually correcting the output of KNP, and CaboCha is a statistical system trained on KC; they are both dependency analyzers. NLPWin, on the other hand, produces a constituent analysis, and was developed independently of KC and its specification. In order to eliminate the factor of the baseline system that seeded the annotation, we have manually annotated 300 newspaper sentences[6] which are not part of KC, but are from the same newspaper database, by hand-correcting the output of NLPWin. In the process of annotation, we tried to follow the KC specification as closely as possible, by referring to the KC annotation guideline (Kurohashi et al., 2000) and using the KC itself as the specification. For this experiment, we obtained precision of 79.29% and recall of 81.04% for NLPWin. These results are slightly but not significantly better than the results on KC, from which we conclude that the baseline system that seeded the annotation does not affect the evaluation results in any significant way. This also confirms the robustness of phrase (bunsetsu)-based annotation in Japanese.

While the effect of the baseline system for annotation may be negligible, there are other factors that introduce spurious differences between the annotated corpus and a system output. In the error analysis we conducted on 200 KC sentences,[7] 7.3% of all errors were due to mapping errors from NLPWin output to KC-style format, caused by the problem of sparseness – that is, even though we tried to follow the KC specification as closely as possible, we always find instances of KC and NLPWin specification differences that were never seen before, therefore not incorporated in the mapping rules. 3.7% of the errors came from sentences that were not properly analyzed because they contained multiple sentences in quotation marks; 5.8% of the errors present ambiguous parses, in which KC and our system defaulted to different correct structures. We also found that 5.1% of the errors were due to annotation errors in KC.

From these figures, it is possible that 20% to 25% of the error cases (i.e., 4% to 5% of all cases) do not actually indicate errors but reflect other differences. This observation is important in interpreting the results for a cross-system comparison in future evaluations.

Scaling Up Phrase-Based Dependency Evaluation

Evaluation of non-newspaper corpora

Given that the proposed evaluation method can scale up to handle different domains of text, it can also be used to compare the performance of a system on different text domains. In order to see this point, we have also annotated 200 sentences each from Microsoft Encarta 98 Encyclopedia and computer manual sentences [8] by manually correcting the output of NLPWin. The annotation was done similarly to the annotation of the 300 newspaper sentences, referring to the same annotation guideline and using the KC annotation as the specification. Except for parentheticals, which needed to be dealt with separately and is discussed in some detail below, the annotation guideline remained perfectly consistent across these domains. The results of running NLPWin on these corpora are shown in Table 2; they match the expectation that the system performs better on these domains than on the newspaper corpus.

corpus	# of sentence	precision	recall
encyclopedia	200	82.81%	83.95%
manual	200	89.93%	91.13%

Table 2: Phrase-dependency based evaluation of NLPWin on encyclopedia and computer manual sentences

Handling parenthetical materials

In creating the annotation for encyclopedia and manual sentences, one difficulty we had in the phrase-based approach was the treatment of parenthetical materials in text. Parentheticals are problematic because they can intervene between content words and function words, disrupting the bunsetsu structure, as in (5):

(5) 欧州連合（ＥＵ）に加盟する。
oushuu_rengou(EU)-ni *kamei-suru*
Europe_union-DAT join-PRES
'join the European Union (EU)'.

In (5), the parenthetical material is inserted between the content word 欧州連合 'European Union' and the dative marker に *ni*. One possible solution to the problem is to ignore all parenthetical materials for the purposes of dependency analysis, treating them as invisible to bunsetsu structure. This is indeed the strategy taken by KC: KC sentences are pre-processed to remove all parenthetical materials; therefore, the annotated sentences include no instances of them. Given that the phrase-based dependency annotation is a lossy method of evaluating the predicate-argument structure to begin with, this approach is not unreasonable. However, information provided by parenthetical materials often provide very useful information for text analysis. For example, the appositive

[5] Average sentence length was 45.39 characters for daily articles and 43.17 for editorial articles.

[6] Consisting of 150 regular and editorial articles each; average sentence length is 45.42 characters.

[7] 100 regular and 100 editorial articles, which are a separate set from the one used in the evaluation for Table 1.

[8] Average sentence length is 49.77 characters for Encarta and 40.31 for the computer manuals.

relation between 欧州連合 'European Union' and EU indicated by parentheses is extremely useful for such tasks as coreference resolution, and are too precious to be excluded from the input text.[9] Also, the majority of parentheticals pose no difficulty for phrase-based dependency annotation, so excluding these cases would be too restrictive.

Since encyclopedia and manual sentences were replete with parenthetical materials, we have chosen to expand the specification of phrase-based annotation to handle parenthetical materials. The specification added to treat parentheticals is summarized in Figure 2, with brackets indicating phrases as necessary. If there are multiple phrases within a set of parentheses, their phrase-breaking and dependency are determined according to the regular specification. The results reported in Table 2 are based on the annotation that followed this expanded specification.

(1) **if** parenthetical is *yomi*, which provides reading information for the preceding character(s)
then do not break into a separate phrase
e.g.: [脆（もろ）さを][感じる]

(2) **else if** parenthetical is within a compound
then do not break into a separate phrase.
e.g.: [Visual Basic for Applications (VBA) プロジェクトが][破損しています]

else

(3) (a) **if** the end of the parenthetical coincides with a phrase-break (i.e., no function word follows the parenthetical)
then the head phrase of the parenthetical modifies the phrase that it semantically modifies. e.g.:
それはありえない（つまり不可能だ）。
それは::ありえない
ありえない::*
つまり::不可能だ
不可能だ::ありえない

(b) **else** (i.e., there are function words that follow the parenthetical)
then wrap the function word(s) with the preceding phrase and have the phrase that immediately precedes the parenthetical modify the head phrase. e.g.:
欧州連合（ＥＵ）に加盟する。
欧州連合::ＥＵに
ＥＵに::加盟する
加盟する::*

Figure 2: Handling of parenthetical materials

In Figure 2, all cases but (3b) are syntactically and semantically transparent, and require no special treatment. (3b) is the only case that presents a dilemma to bunsetsu-based analysis. Though the specification given in Figure 2 may not be ideal and falls short of capturing predicate-argument structure, we believe it is a fair approximation to the same degree as the phrase-based dependency evaluation itself is an approximation of evaluating predicate-argument structure.

Conclusion

Despite the caveats and limitations of the proposed approach, phrase-based dependency evaluation still provides a practical means of evaluating a parser performance, taking advantage of currently available surface-based annotation resources. Though phrase-based dependency evaluation does not measure everything that is relevant to predicate-argument structure (therefore, the method itself has low recall), everything it measures pertains to predicate-argument structure. It is thus excellently suited for tracking system-internal improvements, even of analyzers based on a completely different grammar and linguistic formalism.

Though this paper focused on the evaluation of a Japanese parser, it would certainly be interesting to apply it to different languages, and see how the metric compares to other evaluation methods. It would also be instructive to correlate the accuracy of the proposed approach to the accuracy measurement of predicate argument structure.

Acknowledgements

I would like to thank Mari Brunson for producing the KC-style annotation for various data sets for our experiments.

References

Abney, S.P. 1991. Parsing by Chunks. In R.C. Berwick, S.P. Abney and C. Tenny (eds.), *Principle-Based Parsing: Computation and Psycholinguistics*, pp.257-278. Kluwer Academic Publishers, Boston.

Campbell, R. and H. Suzuki. 2002. Language-Neutral Representation of Syntactic Structure. In *Proceedings of SCANALU-2002*, Heidelberg.

Gee, J.P. and F. Grosjean. 1983. Performance Structures: A Psycholinguistic and Linguistic Appraisal, *Cognitive Psychology 15*, pp.411-458.

Heidorn, G. 2000. Intelligent Writing Assistance. In R.H. Moisl and H. Somers (eds.), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*, Chapter 8, Marcel Dekker, New York.

Kacmarcik, G. 2004. Making Use of Furigana. To Appear in *Proceeding of IJCNLP-04*, Hainan.

Kudo, T. and Y. Matsumoto. 2002. Japanese Dependency Analysis Using Cascaded Chunking. In *Proceeding of CoNLL 2002*, pp.63-69.

Kurohashi, S. and M. Nagao. 1994. KNP parser: Japanese Dependency/Case Structure Analyzer. In *Proceedings of the Workshop on Sharable Natural Language Resources*, pp.48-55.

Kurohashi, S. and M. Nagao. 1997. Kyoto University Text Corpus Project. In *Proceedings of ANLP*, pp.115-118.

Kurohashi, S., Y. Igura and M. Sakaguchi. 2000. Annotation Guideline for Kyoto University Text Corpus (in Japanese).

Lin, D. 1995. A Dependency-Based Method for Evaluating Broad-Coverage Parsers. In *Proceedings of IJCAI-95*, pp.1420-1425.

Lin, D. 1998. Dependency-Based Evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada.

[9] See Kacmarcik (2004) for the use of parenthetical materials in Japanese text.