A Debug Tool for Practical Grammar Development

Akane Yakushiji† **Yuka Tateisi**†‡ **Yusuke Miyao**† **Naoki Yoshinaga**† **Jun'ichi Tsujii**†‡

†Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 JAPAN
‡CREST, JST (Japan Science and Technology Corporation)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012 JAPAN
{akane,yucca,yusuke,yoshinag,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We have developed *willex*, a tool that helps grammar developers to work efficiently by using annotated corpora and recording parsing errors. *Willex* has two major new functions. First, it decreases ambiguity of the parsing results by comparing them to an annotated corpus and removing wrong partial results both automatically and manually. Second, *willex* accumulates parsing errors as data for the developers to clarify the defects of the grammar statistically. We applied *willex* to a large-scale HPSG-style grammar as an example.

1 Introduction

There is an increasing need for syntactical parsers for practical usages, such as information extraction. For example, Yakushiji et al. (2001) extracted argument structures from biomedical papers using a parser based on XHPSG (Tateisi et al., 1998), which is a large-scale HPSG. Although large-scale and general-purpose grammars have been developed, they have a problem of limited coverage.

The limits are derived from deficiencies of grammars themselves. For example, XHPSG cannot treat coordinations of verbs (ex. "*Molybdate* **slowed but did not prevent** *the conversion.*") nor reduced relatives (ex. "*Rb mutants* **derived** *from patients with retinoblastoma.*"). Finding these grammar defects and modifying them require tremendous human effort.

Hence, we have developed *willex* that helps to improve the general-purpose grammars. *Willex* has two major functions. First, it reduces a human workload to improve the general-purpose grammar through using language intuition encoded in syntactically tagged corpora in XML format. Second, it records data of grammar defects to allow developers to have a whole picture of parsing errors found in the target corpora to save debugging time and effort by prioritizing them.

2 What Is the Ideal Grammar Debugging?

There are already other grammar developing tools, such as a grammar writer of XTAG (Paroubek et al., 1992), ALEP (Schmidt et al., 1996), ConTroll (Götz and Meurers, 1997), a tool by Nara Institute of Science and Technology (Miyata et al., 1999), and [incr tsdb()] (Oepen et al., 2002). But these tools have following problems; they largely depend on human debuggers' language intuition, they do not help users to handle large amount of parsing results effectively, and they let human debuggers correct the bugs one after another manually and locally.

To cope with these shortcomings, *willex* proposes an alternative method for more efficient debugging process.

The workflow of the conventional grammar developing tools and *willex* are different in the following ways. With the conventional tools, human debuggers must check each sentence to find out grammar defects and modify them one by one. On the other hand, with *willex* human debuggers check sentences that are tagged with syntactical structure, one by one, find grammar defects, and record them, while *willex* collects the whole grammar defect records. Then human debuggers modify the found grammar defects. This process allows human debuggers to make priority over defects that appear more frequently in the corpora, or defects that are more critical for purposes of syntactical parsing. Indeed, it is possible for human debuggers using the conventional tools to collect and modify the defects but *willex* saves the trouble of human debuggers to collect defects to modify them more efficiently.

3 Functions of *willex*

To create the new debugging tool, we have extended *will* (Imai et al., 1998). *Will* is a browser of parsing results of grammars based on feature structures. *Will* and *willex* are implemented in JAVA.

3.1 Using XML Tagged Corpora

Willex uses sentence boundaries, word chunking, and POSs/labels encoded in XML tagged corpora.

First, with the information of sentence boundaries and word chunking, ambiguity of sentences is reduced, and ambiguity at parsing phase is also reduced. A parser connected to *willex* is assumed to produce only results consistent with the information. An example is shown in Figure 1 (<su> is a sentential tag and <np> is a tag for noun phrases).

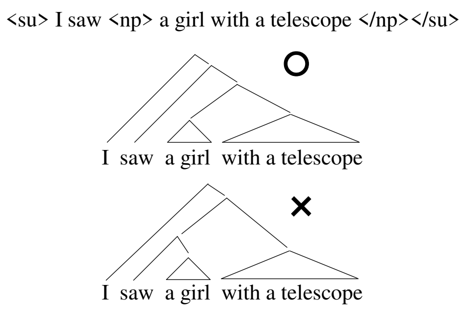

Figure 1: An example of parsing results along with word chunking

Next, *willex* compares POSs/labels encoded in XML tags and parsing results, and deletes improper parsing trees. Therefore, it reduces numbers of partial parsing trees, which appear in the way of parsing and should be checked by human debuggers. In addition, human debuggers can delete partial parsing trees manually later. Figure 2 shows a concrete example. (*NP* and *S* are labels for noun and sentential phrases respectively.)

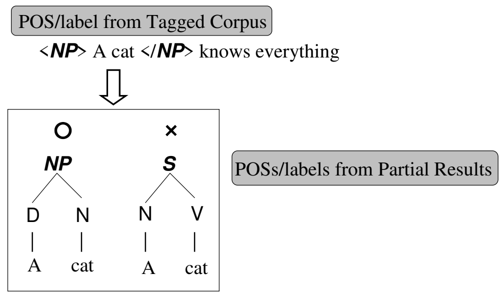

Figure 2: An example of deletion by using POSs/labels

3.2 Output of Grammar Defects

Willex has a function to output information of grammar defects into a file in order to collect the defects data and treat them statistically. In addition, we can save a log of debugging experiences which show what grammar defects are found.

An example of an output file is shown in Table 1. It includes sentence numbers, word ranges in which parsing failed, and comments input by a human debugger. For example, the first row of the table means that the sentence #0 has coordinations of verb phrases at position #3–#12, which cannot be parsed. “OK” in the second row means the sentence is parsed correctly (i.e., no grammar defects are found in the sentence). The third row means that the word #4 of the sentence #2 has no proper lexical entry.

The word ranges are specified by human debuggers using a GUI, which shows parsing results in CKY tables and parse trees. The comments are input by human debuggers in a natural language or chosen from the list of previous comments. A postprocessing module of *willex* sorts the error data by the comments to help statistical analysis.

Table 1: An example of file output

Sentence #	Word #	comment
0	3–12	V-V coordination
1	–	OK
2	4	no lexical entry

4 Experiments and Discussion

We have applied *willex* to *rental-XTAG*, an HPSG-style grammar converted from the XTAG English grammar (The XTAG Research Group, 2001) by a grammar conversion (Yoshinaga and Miyao, 2001).[1] The corpus used is MEDLINE abstracts with tags based on a slightly modified version of GDA-DTD[2] (Hasida, 2003). The corpus is "partially parsed"; the attachments of prepositional phrases are annotated manually.

The tags do not always specify the correct structures based on rental-XTAG (i.e., the grammar assumed by tags is different from rental-XTAG), so we prepared a POS/label conversion table. We can use tagged corpora based on various grammars different from the grammar that the parser is assuming by using POS/label conversion tables.

We investigated 208 sentences (average 24.2 words) from 26 abstracts. 73 sentences were parsed successfully and got correct results. Thus the coverage was 35.1%.

4.1 Qualitative Evaluation

Willex received three major positive feedbacks from a user; first, the function of restricting partial results was helpful, as it allows human debuggers to check fewer results, second, the function to delete incorrect partial results manually was useful, because there are some cases that tags do not specify POSs/labels, and third, human debuggers could use the recording function to make notes to analyze them carefully later.

However, *willex* also received some negative evaluations; the process of locating the cause of parsing failure in a sentence was found to be a bit troublesome. Also, *willex* loses its accuracy if the human debuggers themselves have trouble understanding the correct syntactical structure of a sentence.[3]

We found from these evaluations that the functions of *willex* can be used effectively, though more automation is needed.

4.2 Quantitative Evaluation

Figure 3 shows the decrease in partial parsing trees caused by using the tagged corpus. (Data of 10 sentences among the 208 sentences are shown.) The graph shows that human workload was reduced by using the tagged corpus.

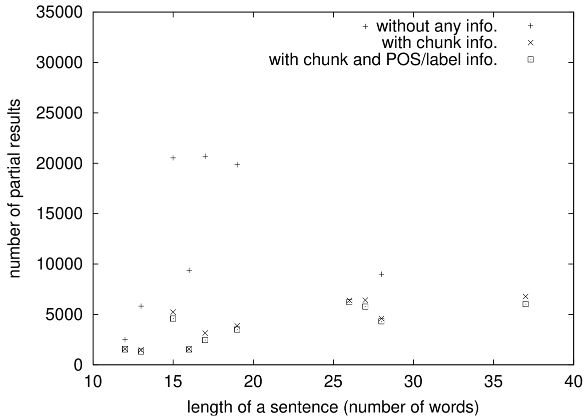

Figure 3: Examples of numbers of partial results

4.3 Defects of rental-XTAG

Table 2 shows the defects of rental-XTAG which are found by using *willex*.

Table 2: The defects of rental-XTAG

the defects of rental-XTAG	#
no lexical entry	62
cannot handle reduced relative	35
cannot handle V-V coordination	22
Adjective does not post-modify NP	9
cannot parse ", but not"	4
cannot handle objective to-infinitive	3
", which ..." does not post-modify NP	3
cannot handle reduced as-relative clause	2
cannot parse "greater than"(">")	2
misc.	17

From this table, it is inferred that (1) lack of lexical entries, (2) inability to parse reduced relative and

[1] Since XTAG and rental-XTAG generate equivalent parse results for the same input, debugging rental-XTAG means debugging XTAG itself.

[2] GDA has no tags which specify prepositional phrases, so we add <prep> and <prepp>.

[3] Thus, we divided the process of identifying grammar defects to two steps. First, a non-expert roughly classifies parsing errors and records temporary memorandums. Then, the non-expert shows typical examples of sentences in each class to experts and identifies grammar defects based on experts' inference. Here, we can make use of the recording function of *willex*.

(3) inability to parse coordinations of verbs are serious problems of rental-XTAG.

4.4 Conflicts Between the Modified GDA and rental-XTAG

Conflicts between rental-XTAG and the grammar on which the modified GDA based cause parsing failures. Statistics of the conflicts is shown in Table 3.

Table 3: Conflicts between the modified GDA and rental-XTAG

modified GDA	rental-XTAG	#
adjectival phrase	verbal phrase	36
bracketing except ","		10
bracketing of ","		8
treatment of omitted words		2
misc.		5

These conflicts cannot be resolved by a simple POS/label conversion table. One resolution is inserting a preprocess module that deletes and moves tags which cause conflicts.

We do not consider these conflicts as grammar defects but the difference of grammars to be absorbed in the conversion phase.

5 Conclusion and Future Work

We developed a debug tool, *willex*, which uses XML tagged corpora and outputs information of grammar defects. By using tagged corpora, *willex* succeeded to reduce human workload. And by recording grammar defects, it provides debugging environment with a bigger perspective. But there remains a problem that a simple POS/label conversion table is not enough to resolve conflicts of a debugged grammar and a grammar assumed by tags. The tool should support to handle the complicated conflicts.

In the future, we will try to modify *willex* to infer causes of parsing errors (semi-)automatically. It is difficult to find a point of parsing failure automatically, because subsentences that have no correspondent partial results are not always the failed point. Hence, we will expand *willex* to find the longest subsentences that are parsed successfully. Words, POS/labels and features of the subsentences can be clues to infer the causes of parsing errors.

References

Thilo Götz and Walt Detmar Meurers. 1997. The ConTroll system as large grammar development platform. In *Proc. of Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 38–45.

Koiti Hasida. 2003. Global document annotation (GDA). available in `http://www.i-content.org/GDA/`.

Hisao Imai, Yusuke Miyao, and Jun'ichi Tsujii. 1998. GUI for an HPSG parser. In *Information Processing Society of Japan SIG Notes NL-127*, pages 173–178, September. In Japanese.

Takashi Miyata, Kazuma Takaoka, and Yuji Matsumoto. 1999. Implementation of GUI debugger for unification-based grammar. In *Information Processing Society of Japan SIG Notes NL-129*, pages 87–94, January. In Japanese.

Stephan Oepen, Emily M. Bender, Uli Callmeier, Dan Flickinger, and Melanie Siegel. 2002. Parallel distributed grammar engineering for practical applications. In *Proc. of the Workshop on Grammar Engineering and Evaluation*, pages 15–21.

Patrick Paroubek, Yves Schabes, and Aravind K. Joshi. 1992. XTAG – a graphical workbench for developing Tree-Adjoining grammars. In *Proc. of the 3rd Conference on Applied Natural Language Processing*, pages 216–223.

Paul Schmidt, Axel Theofilidis, Sibylle Rieder, and Thierry Declerck. 1996. Lean formalisms, linguistic theory, and applications. Grammar development in ALEP. In *Proc. of COLING '96*, volume 1, pages 286–291.

Yuka Tateisi, Kentaro Torisawa, Yusuke Miyao, and Jun'ichi Tsujii. 1998. Translating the XTAG english grammar to HPSG. In *Proc. of TAG+4 workshop*, pages 172–175.

The XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS Research Report 01-03, IRCS, University of Pennsylvania. available in `http://www.cis.upenn.edu/~xtag/`.

Akane Yakushiji, Yuka Tateisi, Yusuke Miyao, and Jun'ichi Tsujii. 2001. Event extraction from biomedical papers using a full parser. In *Pacific Symposium on Biocomputing 2001*, pages 408–419, January.

Naoki Yoshinaga and Yusuke Miyao. 2001. Grammar conversion from LTAG to HPSG. In *Proc. of the sixth ESSLLI Student Session*, pages 309–324.