eBonsai: An integrated environment for annotating treebanks

Ichikawa Hiroshi, Noguchi Masaki, Hashimoto Taiichi, Tokunaga Takenobu, Tanaka Hozumi
Department of Computer Science, Tokyo Institute of Technology
Tokyo Meguro Ôokayama 2-12-1, Japan
ichikawa@cl.cs.titech.ac.jp

Abstract

Syntactically annotated corpora (treebanks) play an important role in recent statistical natural language processing. However, building a large treebank is labor intensive and time consuming work. To remedy this problem, there have been many attempts to develop software tools for annotating treebanks.

This paper presents an integrated environment for annotating a treebank, called eBonsai. eBonsai helps annotators to choose a correct syntactic structure of a sentence from outputs of a parser, allowing the annotators to retrieve similar sentences in the treebank for referring to their structures.

1 Introduction

Statistical approach has been a main stream of natural language processing research for the last decade. Particularly, syntactically annotated corpora (treebanks), such as Penn Treebank (Marcus et al., 1993), Negra Corpus (Skut et al., 1997) and EDR Corpus (Jap, 1994), contribute to improve the performance of morpho-syntactic analysis systems. It is notorious, however, that building a large treebank is labor intensive and time consuming work. In addition, it is quite difficult to keep quality and consistency of a large treebank. To remedy this problem, there have been many attempts to develop software tools for annotating treebanks (Plaehn and Brants, 2000; Bird et al., 2002).

This paper presents an integrated environment for annotating treebanks, called eBonsai. Figure 1 shows a snapshot of eBonsai. eBonsai first performs syntactic analysis of a sentence using a parser based on GLR algorithm (MSLR parser) (Tanaka et al., 1993), and provides candidates of its syntactic structure. An annotator chooses a correct structure from these candidates. When choosing a correct structure, the annotator can consult the system to retrieve already annotated similar sentences to make the current decision. Integration of annotation and retrieval is a significant feature of eBonsai.

To realize the tight coupling of annotation and retrieval, eBonsai has been implemented as the following two plug-in modules of an universal tool platform: Eclipse (The Eclipse Foundation, 2001).

- Annotation plug-in module: This module helps to choose a correct syntactic structure from candidate structures.
- Retrieval plug-in module: This module retrieves similar sentences to a sentence in question from already annotated sentences in the treebank.

These two plug-in modules work cooperatively in the Eclipse framework. For example, information can be transferred easily between these two modules in a copy-and-past manner. Furthermore, since they are implemented as Eclipse plug-in modules, these functionalities can also interact with other plug-in modules and Eclipse native features such as CVS.

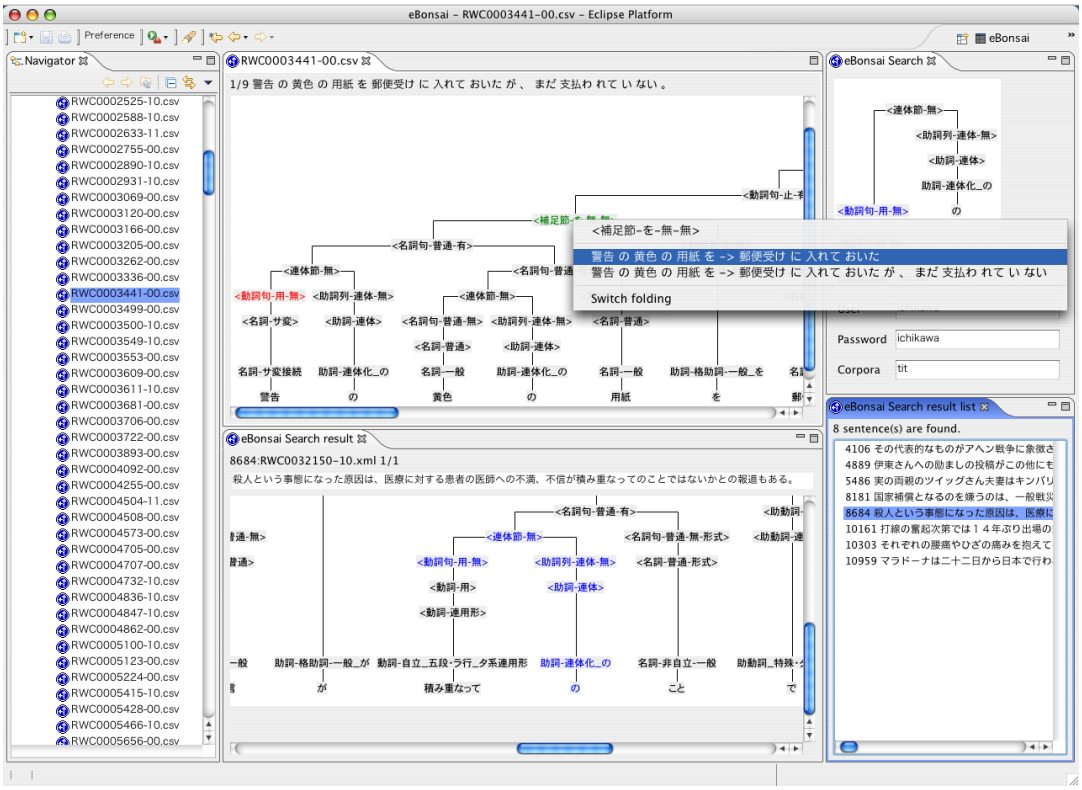


Figure 1: A snapshot of eBonsai

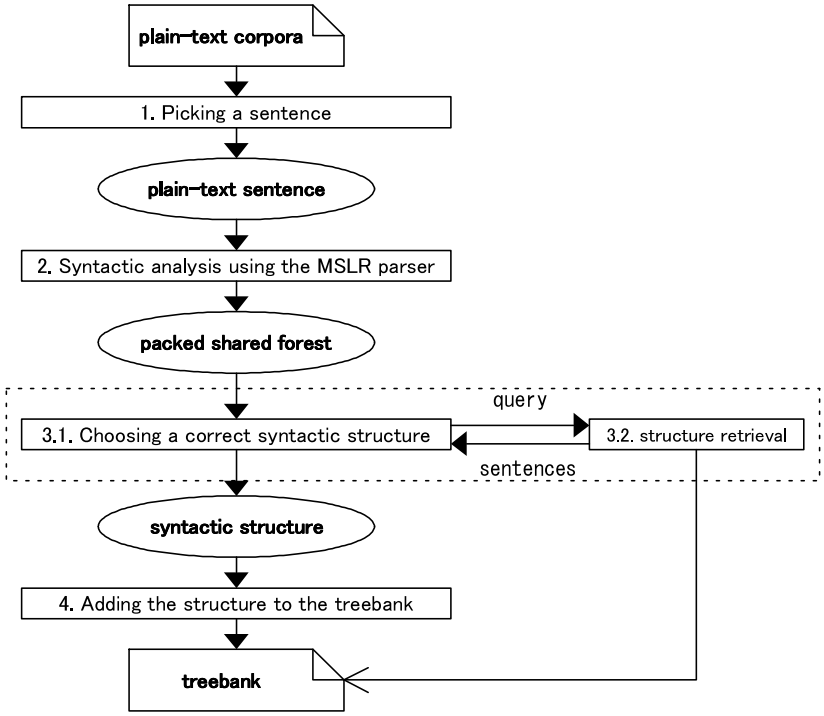


Figure 2: A workflow of annotation using eBonsai

2 Annotating treebanks

Figure 2 shows a workflow of annotating a treebank using eBonsai.

1. An annotator picks a sentence to annotate from plain-text corpora.
2. The MSLR parser (Tanaka et al., 1993) performs syntactic analysis of the sentence.
3. The annotator chooses a correct syntactic structure from the output of the parser. If necessary, retrieval of structures in the treebank is available in this step.
4. The annotator adds the chosen syntactic structure to the treebank.

The coverage of Japanese grammar used in the MSLR parser is fairly wide. The number of grammar rules of the current system is almost 3,000. That means we have a lot of outputs as a result of syntactic analysis in step 2. These structures are represented in terms of a special data structure called packed shared forest (PSF) (Tomita, 1986). The main role of eBonsai is supporting annotators to choose a correct one from a lot of candidate structures in step 3.

3 Annotation plug-in module

3.1 Overview

The annotation plug-in module helps to choose a correct syntactic structure from a set of structures represented by a packed shared forest which is an output of the MSLR parser.

Since there are generally so many syntactic structures given by the parser, it is impractical to find a correct one by checking all of them. The annotation plug-in module shows a single structure at a time as shown in figure 1, and asks annotators to specify a constraint. The system reflects the constraint immediately by filtering out the structures violating it. This process is repeated until a single correct structure is identified. The constraints which can be specified by annotators are following two:

1. Annotators can specify a destination constituent of a dependent constituent.
2. Annotators can specify a correct label of a node.

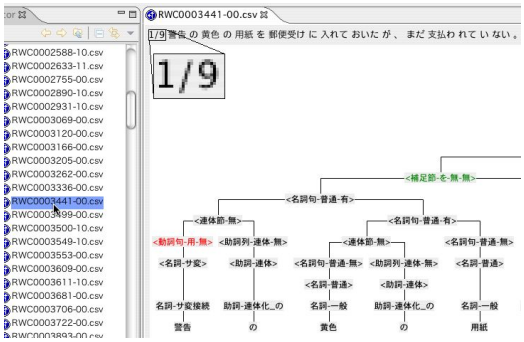
This plug-in module is a reimplementation of an annotation tools developed by Okazaki (Okazaki et al., 2001) in an Eclipse framework.

3.2 Example of annotation

Take the following Japanese sentence for an example to explain the usage of the annotation plug-in module.

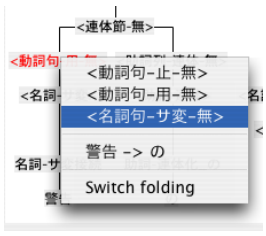
> 警告の黄色の用紙を (yellow paper for warning-ACC) 郵便受けに (mailbox-DAT) 入れておいたが (put, but), まだ支払われていない (not being paid yet).
> *(I put a yellow paper for warning in the mailbox, but it is not paid yet.)*

1. An annotator double-clicks a PSF file name in Eclipse “Navigator” (a left window) to pick up a sentence to annotate.
2. A new window opens and one of the structures is shown in terms of a syntactic tree (a right window).

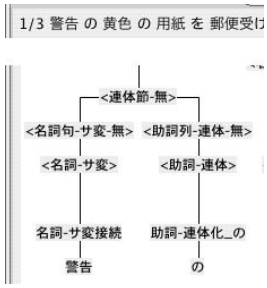


This window is called “Annotation editor”. The notation “1/9” in the left-top part of the window indicates that the presented structure is the first one out of nine.

3. A red node (e.g. “< 動詞句-用-無 >” (verb phrase)) indicates that this node has other possible label names.
4. Clicking a right button on the red node makes a list of possible label names pop up as shown below.

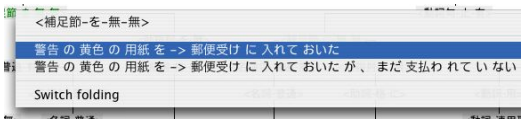


5. Annotators can choose a correct label name of the node in the list. In this case, label “< 名詞句-サ変-無 >” will be selected.
6. Then label “< 動詞句-用-無 >” (verb phrase) changes to “< 名詞句-サ変-無 >” (noun phrase) in the tree and its color becomes black at the same time. Black label names indicate that there is no other possible label for this node.

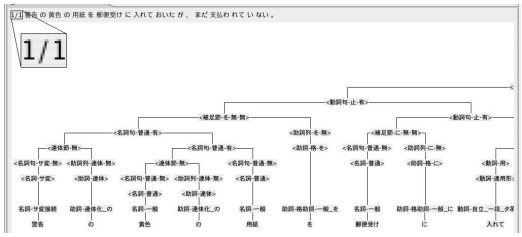


Now, the number of structures shown in the left-top part of Annotation editor decreases to 3.

7. A green node (e.g. “< 補足節-を-無-無 >”) indicates the constituent governed by that node can depend on more than one constituent.
8. Clicking a right button on node “< 補足節-を-無-無 >” makes a list of destinations of dependency pop up as shown below.



9. Annotators can choose a correct destination in the list. In this case, “郵便受けに入れておいた (*put (a yellow paper) in the mailbox*)” will be selected.
10. At this moment, all nodes have turned into black and the number of structure becomes 1. That means the annotation of this sentence has been finished.



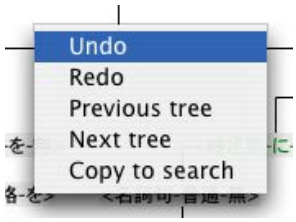


Figure 3: A popup menu of Annotation editor

3.3 Other features

The following features are also implemented.

- Unlimitedly repeatable Undo/Redo. It is possible to undo/redo after saving results by using the popup menu. (figure 3).
- Viewing other structures. Items [Previous tree] and [Next tree] in the popup menu shows different structures.
- Folding constituents. Clicking a right button on a node and selecting item [Switch folding] makes the structure under the node folded. Selecting the same item again unfolds the structure.
- Copying a part of a structure to the retrieval plug-in module. Item [Copy to search] in the popup menu copies a selected structure to the query input window. This feature will be described in detail in the later section.

4 Retrieval plug-in module

4.1 Overview

During the course of annotation, annotators usually put constraints to narrow down to a correct structure considering the meaning of a sentence. However, there are cases in which it is difficult to pin down a correct one by referring to only that sentence. Annotators can consult the system to retrieve the similar structure of sentences in the treebank. The retrieval plug-in module provides annotators such functionality. The retrieval plug-in module receives a syntactic structure as a query and provides a list of sentences which include the given structure.

The retrieval plug-in module has been realized with the method proposed by Yoshida (Yoshida et al., 2004). The method is based on Yoshikawa’s

method (Yoshikawa et al., 2001) which was originally proposed for handling XML documents effectively by using relational database (RDB) systems. Yoshida adopted Yoshikawa's method to deal with syntactic structures in the database. Since an XML document can be represented as a tree, Yoshikawa's method is also applicable to deal with syntactic structures.

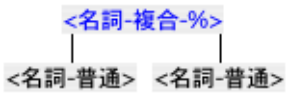


Figure 4: An input query for retrieval

An input query is given as a tree as shown in Figure 4. The structure is then translated into a SQL query and the retrieval is performed.

A query involving a large number of nodes generates a longer SQL query, thus degrades retrieval speed significantly. Yoshida proposed to decompose an input query into a set of subtrees, and to translate each subtree into a SQL query.

4.2 Example of structure retrieval

1. An annotator puts a query tree in the query input window (upper-left window of Figure 5). The query can be modified by the following way.
   - A node label can be changed by clicking a left button on the node and putting a new label in the input area. A label can contain a wild card character "%".
   - A child node can be added by clicking a right button on a node and selecting menu item [Add child].
2. Clicking a right button in the query input window and selecting a menu item starts retrieval. There are four types of search.
   - Menu item [Search] retrieves sentences containing a structure which is exactly the same as the query.
   - Menu item [Partial search] retrieves sentences with less rigid condition than item [Search]. It allows some child nodes missing from the query.

Figure 5: An example of structure retrieval

   - Menu item [Narrow search] searches in the previously retrieved sentences instead of in the entire treebank.
   - Menu item [Partial narrow search] is the combination of [Partial search] and [Narrow search].
3. Retrieval results are shown in the retrieval result list window (a left-bottom window in Figure 5).
4. Clicking a sentence in the list shows the detailed structure of the sentence in the detail window (a right window of Figure 5). A part of the structure matching with the query is colored with blue.
5. If there is more than one substructure matching with the query in a sentence, the system shows the total number of matching parts, and the identifier of the currently colored part by number. Menu items [Previous match] and [Next match] allows annotators to move the other matching parts.

5 Interplay between annotation and retrieval

Since both the annotation plug-in module and the retrieval plug-in module are implemented as

Eclipse plug-ins, they can easily exchange information each other. Thanks for this feature, annotators can copy a part of syntactic structures shown in Annotation editor and submit it to the retrieval module as a query. This can be done by the following procedure.

1. Dragging a mouse pointer over the area covering a target syntactic structure selects the structure, of which color changes to blue.
2. Clicking a right button in Annotation editor makes a command list pop up, and selecting item [Copy to search] copies the selected structure to the query input window.
3. The annotator can modify the query if needed.
4. Clicking a right button in the query input window makes a command list pop up and selecting one of search commands performs a search.

6 Conclusion and Future Work

This paper introduced eBonsai, an integrated environment for annotating treebanks. eBonsai was implemented as two plug-in modules of Eclipse: the annotation plug-in module for choosing a correct syntactic structure from outputs of a parser, and the retrieval plug-in module for retrieving sentences including similar structure to a query in the treebank. These two modules are tightly coupled, thus during the course of annotation, annotators can refer to already annotated sentences in the treebank by using the retrieval module. This helps to annotate difficult cases.

The future work includes the following issues.

- Introduction of a project management functionality by coupling with such as the CVS system.
- Further improvement of the interface.
- Automatic presentation of reference sentences with their structure without annotators' explicit retrieval.
- Functionality to share know-how among annotators.

References

S. Bird, X. Maeda, K. Ma, H. Lee, B. Randall, and S. Zayat. 2002. TableTrans, MultiTrans, InterTrans and TreeTrans: Diverse tools built on the annotation graph toolkit. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 364–370.

Japan Electronic Dictionary Research Institute, 1994. *"Electronic Dictionary User's Manual 2.1"*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

A. Okazaki, K. Shirai, T. Tokunaga, and H. Tanaka. 2001. A syntactic annotation tool with user navigation. In *Proceedings of the 15th Annual Conference of Japanese Society for Artificial Intelligence*.

O. Plaehn and T. Brants. 2000. Annotate – An efficient interactive annotation tool. In *Proceedings of the Sixth Conference on Applied Natural Language Processing ANLP-2000*.

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 88–95.

H. Tanaka, T. Tokunaga, and M. Aizawa. 1993. Integration of morphological and syntactic analysis aased on LR parsing algorithm. In *Proceedings of International Workshop on Parsing Technologies*, pages 101–109.

The Eclipse Foundation. 2001. Eclipse official site. http://www.eclipse.org/.

M. Tomita. 1986. *Efficient Parsing for Natural Language*. Kluwer Academic Publisher.

K. Yoshida, T. Hashimoto, T. Tokunaga, and H. Tanaka. 2004. Retrieving annotated corpora for corpus annotation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1775–1778.

M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. 2001. Xrel: A pathbased approach to storage and retrieval of xml documents using relational database. *ACM Transactions on Internet Technology*, 1(1).