Propbank Instance Annotation Guidelines Using a Dedicated Editor, Jubilee

Jinho D. Choi, Claire Bonial, Martha Palmer

University of Colorado at Boulder
Boulder, CO 80302, USA
choijd@colorado.edu, bonial@colorado.edu, mpalmer@colorado.edu

Abstract

This paper gives guidelines of how to annotate Propbank instances using a dedicated editor, Jubilee. Propbank is a corpus in which the arguments of each verb predicate are annotated with their semantic roles in relation to the predicate. Propbank annotation also requires the choice of a sense ID for each predicate. Jubilee facilitates this annotation process by displaying several resources of syntactic and semantic information simultaneously: the syntactic structure of a sentence is displayed in the main frame, the available senses with their corresponding argument structures are displayed in another frame, all available Propbank arguments are displayed for the annotators choice, and example annotations of each sense of the predicate are available to the annotator for viewing. Easy access to each of these resources allows the annotator to quickly absorb and apply the necessary syntactic and semantic information pertinent to each predicate for consistent and efficient annotation. Jubilee has been successfully adapted to many Propbank projects in several universities. The tool runs platform independently, is light enough to run as an X11 application and supports multiple languages such as Arabic, Chinese, English, Hindi and Korean.

1. Introduction

Jubilee is a Propbank instance editor developed at the University of Colorado at Boulder. Propbank is a corpus in which the arguments of each verb predicate are annotated with their thematic roles (Palmer et al., 2005). In addition, each predicate is annotated with its sense ID, alternatively referred to as a roleset or frameset ID. All the Propbank annotations are done on top of the Penn Treebank style phrase structure (Marcus et al., 1993). For each verb predicate found in every tree of a corpus, we create a Propbank instance that consists of the predicate's sense ID (e.g., `run.02`) and its arguments labeled with numbers corresponding to thematic roles (e.g., `ARG0`, `ARG1`). The argument structure of each individual predicate is outlined in its corresponding frameset file (e.g., `run.xml`). If a tree contains more than one verb predicate, several Propbank instances are generated from the tree. Table 1 shows an example of a Propbank instance associated with a verb predicate, 'run'.

E.g.	John *ran* the Boston Marathon.
`ID`	`run.02` ('walk quickly, a course or contest')
`ARG0`	John ('runner')
`ARG1`	the Boston Marathon ('course, race, distance')

Table 1: Propbank instance associated with 'run'

Prior to annotation, instances that are neither sense-tagged nor annotated are grouped by different predicates and formed into tasks. Each task is double-annotated and subsequently adjudicated; during the adjudication process, the annotation that is the most appropriate is selected for the gold standard. If neither annotation is appropriate, adjudicators may correct the annotation of the instance.

In the past, this procedure was done using several different tools. First, an annotator would claim a task using a command-line tool. Each task could be claimed by only a certain number of annotators, generally two, to prevent annotation of the same task by more annotators than intended. After claiming a task, the annotator would start annotation using a different tool with a GUI. Using this tool, the annotator could view the tree structure of each instance in parenthetical notation. However, visualizing the constituent boundaries and selecting the appropriate node for an argument was often very difficult for annotators with less expertise in this format. Furthermore, the tool did not support information from frameset files, which must be consulted for the appropriate annotations. Once the task was completely double-annotated, an adjudicator would adjudicate the annotations using the same tool; however, the adjudicator would subsequently use another tool to denote the verb sense of each instance. Thus, three different tools were required to create one Propbank instance. Although each tool worked effectively on its own, using several tools for one task not only decreased the efficiency of the project, but also required another process of checking if all tools were used properly. This motivated the creation of a new annotation tool, Jubilee, which can do all of the above and more.

With Jubilee, the entire annotation procedure can be done using one tool that simultaneously provides rich syntactic information as well as comprehensive semantic information. More importantly, the use of Jubilee ensures that all the annotations are saved in one place using one unified format, the lack of which had been a delaying factor when using other tools. Moreover, this makes data maintenance much easier and more consistent. The tool is highly adaptable such that creating a new Propbank project from a different corpus is as easy as editing a simple text file.

Jubilee is developed in Java (JDK 6.0), so it runs on any platform where a Java virtual machine is installed. It is light enough to run as an X11 application. This aspect is important because Propbank files are usually stored in a server, so annotations need to be done remotely (via SSH). One of the biggest advantages of using Jubilee is that it accommo-

dates several different languages; in fact, it has been used for Propbank projects in Arabic (M.Diab et al., 2008), Chinese (Xue and Palmer, 2009), English (Palmer et al., 2005), and has been tested in Hindi and Korean (Han et al., 2002).

This paper details how to annotate Propbank instances using Jubilee. There are two modes in which to run Jubilee: normal and gold mode. In normal mode, annotators are allowed to view and edit only tasks that have been claimed by themselves or by one other annotator. In gold mode, adjudicators are allowed to view and edit all tasks that have undergone at least single-annotation. Annotators and adjudicators are expected to run Jubilee in normal and gold mode, respectively. Although there are two different modes, the interfaces are very similar, so learning one mode effectively teaches the other.

2. How to obtain Jubilee

Jubilee is available as an open source project on Google code.[1] The webpage gives detailed instructions of how to download, install and launch the tool (Choi et al., 2009).

3. Jubilee in normal mode

Annotators are expected to run Jubilee in normal mode. In normal mode, annotators are allowed to view and edit only tasks claimed by themselves or one other annotator when the max-number of annotators allowed is two. Jubilee gives the option of assigning a different max-number of annotators as well.

When you run Jubilee in normal mode, you will see an open-dialog (Figure 1). There are three components in the open-dialog. The combo-box at the top shows a list of all Propbank projects. Once you select a project (e.g., `english.sample`), both `[New Tasks]` and `[My Tasks]` will be updated. `[New Task]` shows a list of tasks that have either not been claimed, or claimed by only one other annotator. `[My Tasks]` shows a list of tasks that have been claimed by the current annotator.

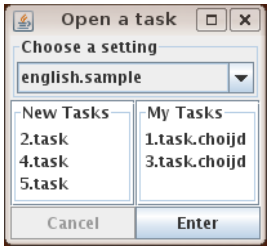


Figure 1: Open-dialog

Once you choose a task and click the `[Enter]` button, Jubilee's main window will be prompted (Figure 5). There are three views available in the main window: the treebank view, frameset view and argument view. The following sections explain how each view is used for Propbank annotation.

3.1. Treebank view

By default, the treebank view shows the first tree in the selected task. Each node in the tree consists of two elements: POS-tag and word-token. Upon annotation, a Propbank argument label is added to each node constituting an argument of the selected predicate. The verb predicate is marked with a special tag '`rel`'. The text-box on the right shows the user ID indicating the user annotating this instance. The text-box at the bottom shows the raw sentence of the tree.

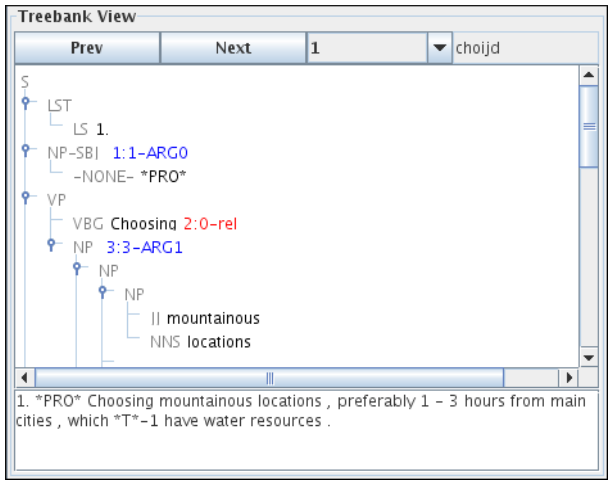


Figure 2: Treebank view

3.2. Frameset view

The frameset view displays and allows the annotator to choose the sense of the predicate with respect to the current tree. The predicate text-box on the left shows the lemma of the predicate associated with the selected roleset.[2] The roleset combo-box at the middle gives the full list of roleset IDs associated with the predicate lemma. As a roleset is selected, a short definition of the roleset as well as its generalized argument structure appear in the roleset information pane. To view annotation examples of the currently selected roleset, click the `[Example]` button.

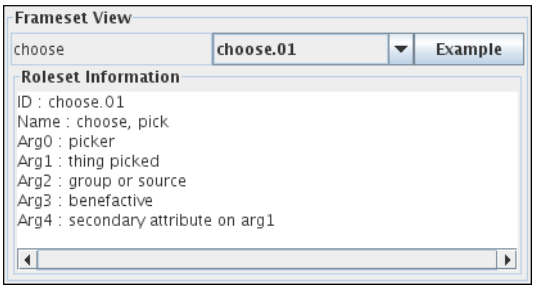


Figure 3: Frameset view

3.3. Argument view

The argument view contains buttons representing each of the Propbank argument labels. To annotate an argument (e.g., `ARG0`), select the node to be annotated in the treebank view, then click the corresponding argument button (e.g., `[0]`). When you click the button, the argument label

[1] `http://code.google.com/p/propbank/`

[2] The term 'roleset' is interchangeable with a term 'frameset' depending on the language being annotated.

appears on the selected node, along with its relative location in the tree (e.g., `1:1`). The first number of this designation indicates the index of the word-token closest to the selected node and the second number indicates the phrase level of the node from the closest word-token. The `[Erase]` button is used to remove the annotation and the `[-UNDEF]` button is used to assign an argument that is not defined in the argument structure of the currently selected roleset.

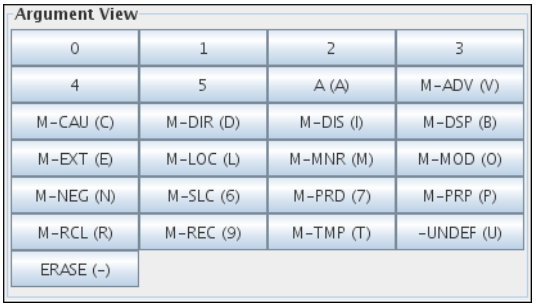


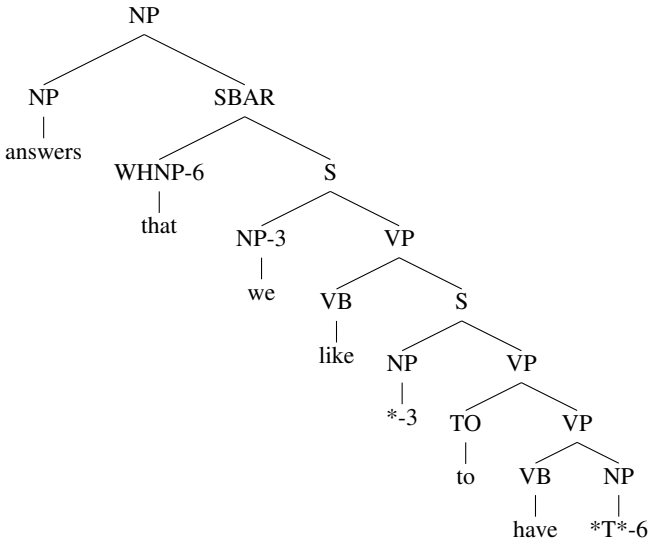
Figure 4: Argument view

3.4. Links and concatenations

Several operators are used to perform concatenations and coreference links. For example, the '⋆' operator is used to link relative pronouns to their antecedents, which are never co-indexed in the Treebank.

Propbank:
`REL` : have
`ARG0`: [NP *-3]
`ARG1`: [NP *T*-6]
`LINK-SLC`: [WHNP-6 that] ⋆ [NP answers]

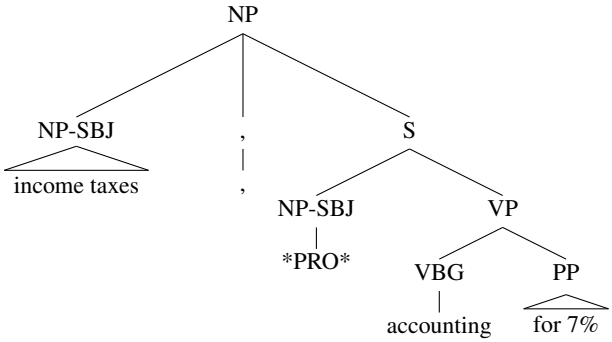
Treebank:



In the absence of Treebank co-indexing between a null element and its overt referent, annotators can provide semantic information about the null element by manually linking it to its overt referent using the '⋆' operator.

Propbank:
`REL` : accounting
`ARG0`: [NP-SBJ *PRO*] ⋆ [NP-SBJ income taxes]
`ARG1`: [PP for 7%]

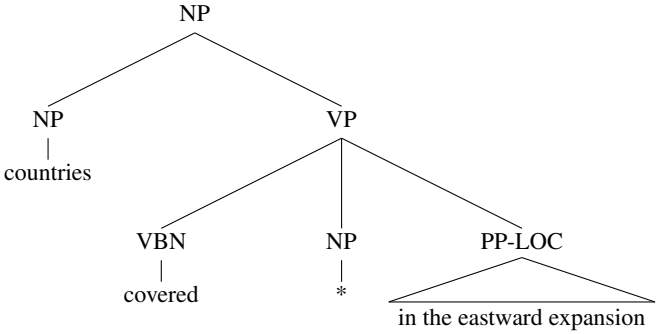
Treebank:



Similarly, the '&' operator is used to link the object trace after a passive verb to its referent in the subject position in reduced relative constructions.

Propbank:
`REL` : covered
`ARG1`: [NP *] & [NP countries]
`ARGM-LOC`: [PP-LOC in the eastward expansion]

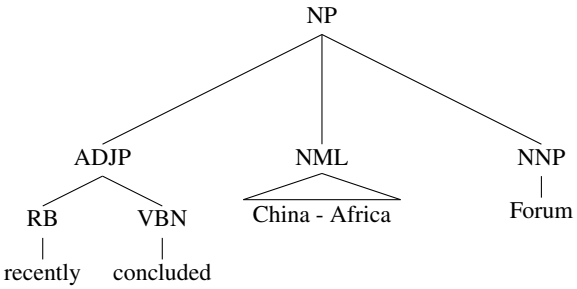
Treebank:



Additionally, in the cases where an argument is discontinuous such that it cannot be captured in the annotation of one node, the ',' operator is used to concatenate more than one node into a single argument.

Propbank:
`REL` : concluded
`ARG1`: [NML China - Africa] , [NNP Forum]
`ARGM-TMP`: [RB recently]

Treebank:



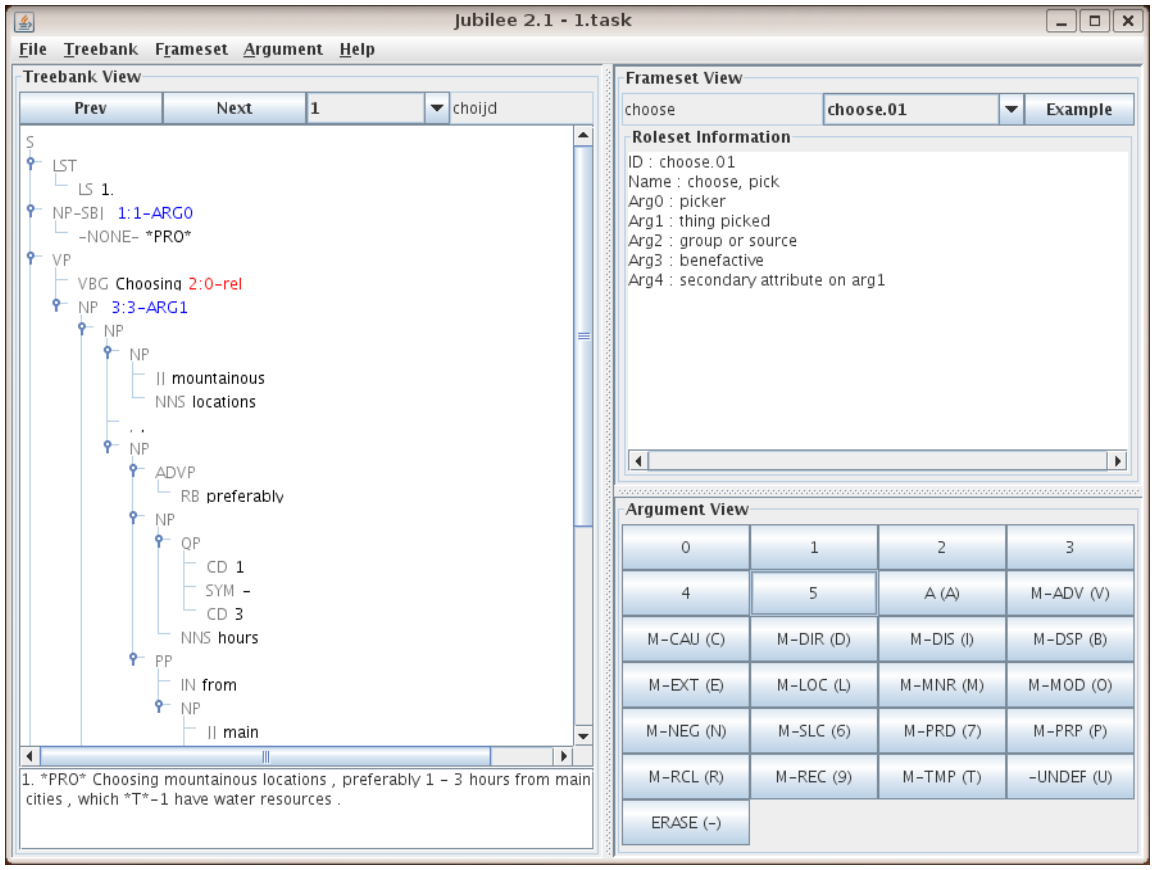


Figure 5: Jubilee's main window

Thus, each of the described operators assists annotators in capturing the appropriate semantic information given certain syntactic constraints.

3.5. Verb particle constructions

Some English verbs use particles to form their predicate lemmas (e.g., 'run out', 'run up'). For such verb particle constructions, the particle is concatenated with the verb to form a single predicate lemma, annotated with the '`rel`' ('relation') tag. To concatenate the particle, choose the particle node and type `Ctrl+Shift+/`. The resulting `rel` annotation will reflect the locations of both the original predicate and the concatenated particle. Note that if the annotator accidentally erases the `rel` tag on any verb, it can be recovered using the same shortcut, `Ctrl+Shift+/`, on the verb.

4. Jubilee in gold mode

Adjudicators are expected to run Jubilee in gold mode. In gold mode, adjudicators are allowed to view and edit all tasks that have undergone at least single-annotation. When you run Jubilee in gold mode, you will see the same open-dialog as you saw in Figure. 1. The `[New Tasks]` shows a list of tasks that have not been adjudicated and the `[My Tasks]` shows a list of tasks have been adjudicated. Gold mode does not allow adjudicators to open tasks that have not been at least single-annotated.

Once you launch Jubilee in gold mode, you will see almost the same main window as in Figure 5, except there will be a list showing all annotations for the currently selected instance at the top of the treebank view (Figure 6). The first line shows the annotation done by the adjudicator, and the following lines show annotations done by different annotators. By default, the first annotation is chosen as gold. The adjudicator can choose an alternate annotation by clicking on the other annotation. The treebank view dynamically changes to reflect the selected annotation. Jubilee also gives an option of skipping instances for which all annotations are the same, thereby quickening the pace of adjudication. To modify the annotation in cases where neither is perfectly correct, the adjudicator can simply manipulate the arguments as described for the normal mode.

5. Software demonstration

For the treebank view, we will compare its graphical representation of the trees with the parenthetical representation: the clear visual representation of the phrase structure helps the annotator to better understand the syntax of the instance and to annotate the appropriate node within the correct span. For the frameset view, we will detail what kind of semantic information it provides as you choose different rolesets. This will highlight how Jubilee's support of sense annotation not only speeds up the annotation process, but also ensures consistent annotation because the roleset information provides a guideline for the correct annotation of a particular verb sense. For the argument view, we will illustrate how to annotate Propbank arguments and use the operators for concatenations and links; thereby also demon-

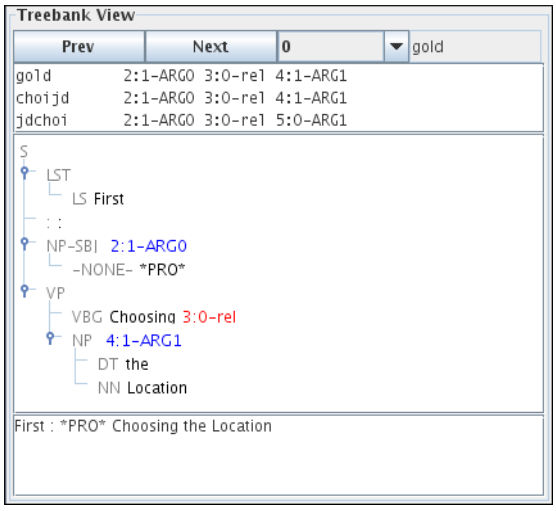


Figure 6: The treebank view in gold mode

strating that having each of these labels clearly visible helps the annotator to remember and evaluate the appropriateness of each possible argument label. Finally, we will show how intuitive it is to adjudicate the annotations in gold mode.

6. Advantages and future work

Jubilee has drastically simplified and streamlined the Propbank annotation process because what was once done with three separate tools can now be completed with one. This has several distinct advantages. First, it speeds up the entire process of annotation and adjudication because argument annotation and the choice of a sense ID (roleset or frameset ID) for the verb can be completed simultaneously. This also ensures that annotators have a greater awareness of the argument structure and semantics of the predicate as defined in the frameset file. Thus, annotators can quickly and efficiently familiarize themselves with a predicate and apply that knowledge consistently to each instance. Additionally, Jubilee's treebank view gives the annotator a more effective visual representation of the syntax of each instance than the previous parenthetical notation, clarifying the relationship between the argument structure found in the frame and that of the individual instance. Finally, the use of one tool simplifies data maintenance because one unified file format can be used throughout the process.

Jubilee has been successfully adapted to Propbank projects in several universities such as the University of Colorado at Boulder, the University of Illinois at Urbana-Champaign, and Brandeis University. We will continuously develop the tool by improving its functionalities through user-testing and feedback, and also by applying it to more languages.

Acknowledgments

Special thanks are due to Professor Nianwen Xue of Brandeis University for his very helpful insights, as well as Scott Cotton, the developer of RATS and Tom Morton, the developer of WordFreak, both previously used for PropBank annotation.

We also gratefully acknowledge the support of the National Science Foundation Grants CISE-CRI-0551615, Towards a Comprehensive Linguistic Annotation and CISE-CRI 0709167, Collaborative: A Multi-Representational and Multi-Layered Treebank for Hindi/Urdu, and a grant from the Defense Advanced Research Projects Agency (DARPA/IPTO) under the GALE program, DARPA/CMO Contract No. HR0011-06-C-0022, subcontract from BBN, Inc. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. References

Jinho D. Choi, Claire Bonial, and Martha Palmer. 2009. Jubilee: Propbank instance editor guideline (version 2.1). Technical report, Institute of Cognitive Science, the University of Colorado at Boulder.

C. Han, N. Han, E. Ko, and M. Palmer. 2002. Korean treebank: Development and evaluation. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

M.Diab, A.Mansouri, M.Palmer, O.Babko-Malaya, W Zaghouani, A.Bies, and M.Maamouri. 2008. A pilot arabic propbank. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Nianwen Xue and Martha Palmer. 2009. Adding semantic roles to the chinese treebank. *Natural Language Engineering*, 15(1):143–172.