Propbank Frameset Annotation Guidelines Using a Dedicated Editor, Cornerstone

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 create and update Propbank frameset files using a dedicated editor, Cornerstone. 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. Thus, for each predicate in Propbank, there exists a corresponding frameset file showing the expected predicate argument structure of each sense related to the predicate. Since most Propbank annotations are based on the predicate argument structure defined in the frameset files, it is important to keep the files consistent, simple to read as well as easy to update. The frameset files are written in XML, which can be difficult to edit when using a simple text editor. Therefore, it is helpful to develop a user-friendly editor such as Cornerstone, specifically customized to create and edit frameset files. Cornerstone 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

Cornerstone is a Propbank frameset 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). For each predicate in the Propbank, there exists a corresponding frameset file encompassing one or more senses of the predicate. For example, for a verb predicate 'run', there exists a frameset file, `run.xml`, that describes verb-senses of the predicate (e.g., `run.01`). Additional senses can be added to the frameset file as they arise in the Propbank. For English, in addition to senses corresponding to the main predicate lemma (e.g., 'run'), a frameset file may also include senses corresponding to any verb particle constructions associated with the predicate (e.g., 'run out', 'run up').

Each sense, alternately referred to as a roleset or a frameset depending on the language, comes with a generalized predicate argument structure of the sense as well as annotated examples from the corpus. For example, a sense `run.02` ('walk quickly, a course or contest') lists three roles represented by numbered arguments: `ARG0` as a 'runner', `ARG1` as a 'course, race or distance', and `ARG2` as an 'opponent'. The frameset file is essential for Propbank annotation because it not only supplies semantic information about each sense, but also defines the predicate argument structure of the sense, providing guidelines as to how that particular sense should be annotated. Since Propbank annotations are based on the argument structure outlined in the frameset files, it is important to keep them consistent, simple to read, and easy to update.

All frameset files are written in XML, which provides a useful, hierarchical format suited to the project. However, the format is somewhat complicated to read, especially for the annotators and adjudicators who are not familiar with it. Most importantly, it is difficult to edit XML files using a simple text editor without making mistakes that could be detrimental to the operation of the project. Therefore, it is helpful to have a user-friendly editor to create, view, and edit frameset files without knowing about XML. Although many XML editors already exist, most of them require some degree of knowledge of XML, and none of them are specifically customized for frameset files. This motivated the development of our own frameset editor, Cornerstone.

Cornerstone 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 frameset files are usually stored in a server, so frame authors need to update the files remotely (via SSH). One of the biggest advantages of using Cornerstone is that it accommodates several languages; in fact, the tool has been used for Propbank projects in Arabic (M.Diab et al., 2008), Chinese (Xue and Palmer, 2009), English (Palmer et al., 2005) and Hindi, and has been tested in Korean (Han et al., 2002).

This paper details how to create and update the frameset files using Cornerstone. There are two modes in which to run Cornerstone: multi-lemma and uni-lemma mode. In multi-lemma mode, a predicate can have multiple lemmas, whereas a predicate can have only one lemma in uni-lemma mode. Languages such as English and Hindi are expected to run in multi-lemma mode, and languages such as Arabic and Chinese are expected to run in uni-lemma mode. Although there are two different modes, the interfaces are very similar, so learning one mode effectively teaches the other.

2. How to obtain Cornerstone

Cornerstone 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).

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

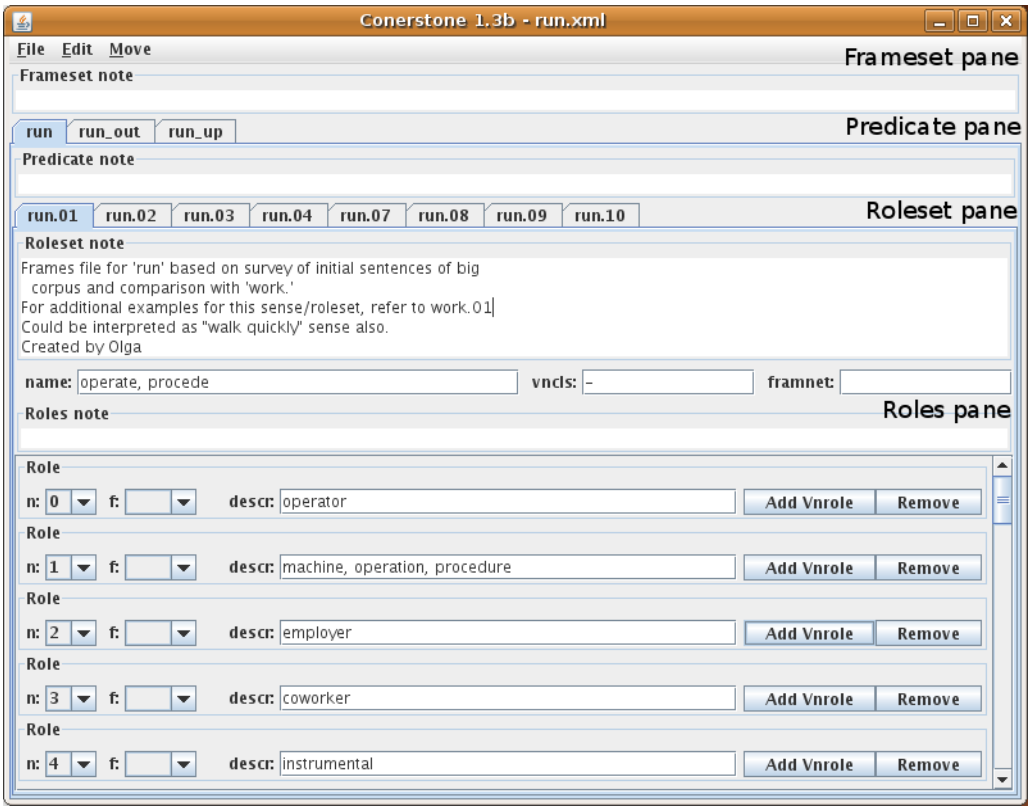

Figure 1: Open `run.xml` in multi-lemma mode

3. Cornerstone in multi-lemma mode

Languages such as English and Hindi are expected to run in *multi-lemma* mode, due to the nature of their verb predicates. In multi-lemma mode, a predicate can have multiple lemmas (e.g., 'run', 'run out', 'run up'). The XML structure of the frameset files for such langauges is defined in a DTD file, `frameset.dtd`.

Figure 1 (Page 2) shows what appears when you open a frameset file, `run.xml`, in multi-lemma mode. The window consists of four panes: the frameset pane, predicate pane, roleset pane and roles pane. The frameset pane contains the predicate pane as well as a frameset note reserved for information that pertains to all predicate lemmas and rolesets within the frameset file.

The predicate pane contains one or more tabs titled by predicate lemmas that may include verb particle constructions. Each predicate tab contains a roleset pane as well as a predicate note for optional information that pertains to all rolesets encompassed by the predicate lemma.

The roleset pane contains tabs titled by roleset IDs (e.g., `run.01`, `run.02`) for the currently selected predicate lemma (e.g., 'run'). Each roleset tab contains a roleset note for required information about the sense. This information includes, but is not limited to, the corpus that is the source of that roleset, the VerbNet class (Kipper et al., 2006) that the predicate falls into (or a note that VerbNet does not include the particular predicate) and the author of the roleset. Optional information may be a mention of other predicates that were consulted in comparison to the current predicate (especially in the absence of a VerbNet entry) or relevant FrameNet information (Baker et al., 1998). [2] The roleset pane also contains three attribute fields: `name`, `vncls`, and `framnet`. The `name` attribute shows a brief definition of the current roleset. The `vncls` and the `framnet` attributes show which VerbNet and FrameNet class this roleset is associated with, respectively. These `vncls` and the `framnet` attributes are useful because PropBank's verb senses are very coarse-grained: a new sense is only added when both the syntax and semantics of a new usage differ from an existing roleset. Thus, the mappings to VerbNet and FrameNet classes can be used to supplement information about what finer-grained verb senses correspond to the PropBank roleset. Additionally, the roleset pane contains a roles pane.

The roles pane includes one or more verb-specific roles, which represent arguments that the predicate requires or commonly takes in usage. For example, a roleset `decrease.01` has roles representing five arguments: `ARG0` as a 'causer of decline, agent', `ARG1` as a 'thing decreasing', `ARG2` as an 'amount decreased by', `ARG3` a 'starting point' and `ARG4` as an 'ending point'. Each role contains three attribute fields: `n` is an argument number, `f` is a function tag and `descr` shows a description of the role. The relationship between argument numbers, `n`, and thematic roles is intended to be somewhat flexible and can be changed across different predicates. However, numbered arguments generally correspond to the following thematic roles in Table 1.

The function tag, `f` is available for each role. If the

[2] VerbNet and FrameNet information is not currently linked in languages other than English.

ARG0	agent	ARG3	starting point
ARG1	patient	ARG4	ending point
ARG2	instrument benafactive attribute	ARGM	modifier

Table 1: List of arguments in Propbank

survey of a given predicate shows that a certain type of modifier (e.g., locative, temporal) is commonly used with the predicate, then the frame author can add a role labeled ARGM with the appropriate function tag (e.g., loc, tmp; see Palmer et al. (2005)) in place of a numbered argument. The attribute field descr contains a description of the semantic role, general enough to be applied to various syntactic realizations of this role (e.g., ARG0 for run.02 is a 'runner').

Each role can include vnrole (VerbNet role) information. There are two attribute fields within vnrole: vncls (VerbNet class) and vntheta (VerbNet thematic role). If the predicate is a member of VerbNet, this information should be supplied for each role compatible with the VerbNet information (for more details about VerbNet, see Kipper et al. (2006)). The VerbNet class is the larger group of verbs of which the predicate in question is a member. These classes are numbered, and also named, generally with a verb that is a canonical member of this class. For example, 'run' is a member of several VerbNet classes, including BUMP-18.4, CARRY-11.4, MEANDER-47.7, RUN-51.3.2; the earlier example run.02 is mapped only to the relevant class RUN-51.3.2. The second attribute, vntheta gives the VerbNet thematic role correlated with the Propbank role. For example, the ARG0 of run.02 is correlated with the vntheta 'agent'.

Not only is the VerbNet information useful to annotators who find it helpful to view the more canonical thematic role associated with a numbered argument, this information can also supplement potential weaknesses of PropBank's verb-specific arguments. Although PropBank's ARG0 and ARG1 consistently correspond to prototypical agents and patients respectively, ARG2 through ARG5 are highly variable. However, the mappings between PropBank's numbered arguments and VerbNet thematic roles can be used to provide consistent thematic role labels across several verbs.

Like other panes, the roles pane also contains a roles note for optional information about the roles. This may include information that will help annotators disambiguate between roles and some syntactic information relevant to the roles.

4. Cornerstone in uni-lemma mode

Languages such as Arabic and Chinese are expected to run in *uni-lemma* mode. Unlike multi-lemma mode, which allows a predicate to have multiple lemmas, uni-lemma mode allows only one lemma for a predicate. The XML structure of the frameset files for such langauges is defined in a DTD file, verb.dtd.

Figure 2 (in page 4) shows what appears when you open a frameset file, HAfaZ.xml, in uni-lemma mode. The window consists of four panes: the verb pane, frameset pane, frame pane and roles pane. The verb pane contains a verb comment for helpful information about the verb, and an attribute field, ID, indicating the predicate lemma of the verb, which can be represented either in Roman alphabets or characters in other languages. Additionally, the verb pane contains the frameset pane.

The frameset pane contains several tabs titled by frameset IDs for the predicate. Note that the frameset in uni-lemma mode is equivalent to the roleset in multi-lemma mode. The frameset pane also contains a frameset comment for required information about the currently selected frameset as well as two attribute fields, edef and cdef, which show the English and non-English (in this case, Arabic) definitions of the frameset, respectively. In addition, the frameset pane contains one or more frame panes and the roles pane.

The frame pane contains a frame comment for optional information about the frame and the mapping pane. In turn, the mapping pane contains a comment used to describe mappings between syntactic and semantic arguments associated with the currently selected frameset. The mapping pane also contains V, which is a placeholder indicating where the verb predicate should be located among the other arguments, and a set of mappings between each syntactic argument, src (e.g., subject, object) and a semantic argument, trg (e.g., agent, patient). The syntactic arguments are often provided in the Treebank (Xue and Palmer, 2009), in which case, these mappings can be used for automatic extraction of semantic arguments from their syntactic labels. Table 2 shows the full list of syntactic arguments.

Argument	**Description**
sbj	subject
npobj	noun-phrase object
ipobj	inflectional-phrase object
ext	extent
dir	direction
controlip	ipobj that is a control clause
io	indirect object
other	other kind of syntactic arg.

Table 2: List of syntactic arguments

The roles pane consists of a set of arguments that the predicate requires or commonly takes. Each argument has two attribute fields: argnum is an argument number (Table 1) and argrole shows a description of the semantic role.

5. Software demonstration

We will begin by demonstrating how to view frameset files in both multi-lemma and uni-lemma mode. In each mode, we will open an existing frameset file, compare its interface with the actual XML file, and show how intuitive it is to interact with the tool. Next, we will demonstrate how to create and edit a new frameset file either from scratch or by copying from a related frameset file. This demonstration will reflect several advantages of using the tool. First, the XML structure is completely transparent to the frame

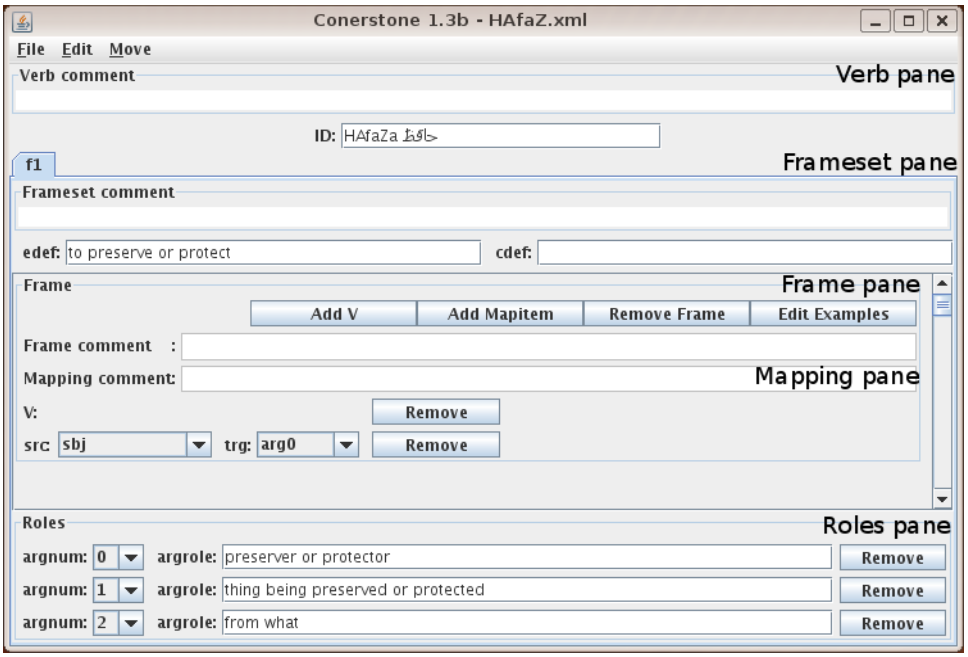

Figure 2: Open `HAfaZ.xml` in uni-lemma mode

authors, so that no knowledge of XML is required to manage the frameset files. Second, the tool automates some of the routine work for the frame authors (e.g., assigning a new roleset/frameset ID) and gives lists of options to be chosen (e.g., a list of function tags) so that frameset creation, and the entire annotation procedure in turn, become much faster. Third, the tool checks for the completion of required fields and formatting errors so that frame authors do not have to check them manually. Finally, the tool automatically saves the changes so the work is never lost.

6. Advantages and future work

Since Propbank annotations are based on the frameset files, it is important to keep them consistent as well as easy to update. The frameset files are written in XML, which is difficult to edit using a simple text editor. By using Cornerstone, you can view, create and edit frameset files without knowing XML. Furthermore, the frameset files created by Cornerstone are guaranteed to be free of the errors that commonly occur when directly manipulating XML files.

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

Acknowledgments

We 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

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*.

Jinho D. Choi, Claire Bonial, and Martha Palmer. 2009. Cornerstone: Propbank frameset editor guideline (version 1.3). 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*.

K. Kipper, A.Korhonen, N.Ryant, and M.Palmer. 2006. Extending verbnet with novel verb classes. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

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.