Cornerstone: Propbank Frameset Editor Guideline (Version 1.3)

Jinho D. Choi
choijd@colorado.edu

Claire Bonial
bonial@colorado.edu

Martha Palmer
mpalmer@colorado.edu

Center for Computational Language and EducAtion Research
Institute of Cognitive Science
University of Colorado at Boulder

Institute of Cognitive Science
Technical Report 01-09

September 28, 2009

Propbank: córpus com SRL

Cornestone: ferramenta para criar framefile (estruturar a sentença para processar língua)

possíveis sentidos do verbo = verb senses; frameset ou roleset ID

Abstract

This report gives a guideline of how to use a Propbank frameset editor, Cornerstone. Propbank is a corpus where 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 (also known as a frameset or roleset ID) for each predicate. Therefore, for each predicate in the Propbank, there exists a corresponding frameset file that shows the predicate argument structure of all senses 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. Up to this point, all frameset files are written in XML format, which provides a well-organized hierarchical structure, but is difficult to edit without making mistakes. Therefore, it is necessary to develop a user-friendly editor such as Cornerstone, that is specifically customized to view, 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.

tornar tudo mais fácil para o usuário. O Cornerstone já faz isso, mas podemos melhorar?

Contents

1 Introduction

Cornerstone is a Propbank frameset editor developed at the University of Colorado at Boulder. Propbank is a corpus where the arguments of each verb predicate are annotated with their semantic roles in relation to the predicate (Palmer et al., 2005). For each predicate appearing 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 each of the established senses of the predicate (e.g., `run.01`, `run.02`). Additional senses can be added to the frameset file as they arise in the Propbank. Note that these senses are not as fine-grained as the ones in WordNet (Fellbaum, 1998). Only if the usage of a predicate is both semantically and syntactically unique will it constitute a new sense. 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 in the frameset file, also known as roleset or frameset depending on the languages, includes a generalized predicate argument structure of the sense as well as its annotated examples from the corpus.[1] For example, a sense `run.02` ('walk quickly, a course or contest') comes with three roles listed as numbered arguments: `ARG0` as a 'runner', `ARG1` as a 'course, race or distance', and `ARG2` as an 'opponent'. Thus, 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, which gives a guideline as to how that particular sense should be annotated. Since most Propbank annotations are based on the frameset files, it is important to keep the files consistent, simple to read as well as easy to update. Note that for English, the predicate argument structure illustrated in the frameset file is intended to be compatible with the thematic roles outlined by VerbNet (Kipper et al., 2006). There are cases, however, in which a VerbNet entry does not exist for the Propbank predicate, or the thematic roles associated with the predicate's VerbNet class simply do not fit with the usage found in the Propbank corpus. Nonetheless, the sense should ideally provide information about which VerbNet class the predicate falls into, and include mappings between each role and the corresponding VerbNet thematic role.

XML é bom para o computador, mas não para o humano anotador

All frameset files are written in Extensible Markup Language (XML) format. XML provides a useful, hierarchical format that is suited to the project. However, the format is somewhat complicated to read, especially for annotators and adjudicators who are not familiar with the language. Most importantly, it is difficult to edit XML files using some kind of text editor without making mistakes, which can cause further ramifications on the operation of the project. Therefore, it is necessary to have a user-friendly editor to view, create and edit frameset files without knowing much about XML. Although many XML editors already exist, most of them require some 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, J2SE Development Kit (JDK) 6.0, so it runs on all kinds of platforms (Microsoft Windows, Mac OS and Linux) as long as the right version of JDK is installed.[2] It is light enough to run as an X11 application. This aspect is important because frameset files are usually stored in a server, and annotators are to update the files remotely (via SSH) by using their local machines. One of the biggest advantages of using Cornerstone is that it runs on many different languages; in fact, the tool has been used for Propbank projects in Arabic (Diab et al., 2008), Chinese (Xue and Palmer, 2009), English (Palmer et al., 2005) and Hindi, and also has been tested in Korean (Han et al., 2002).

This report details how to setup and run Cornerstone in multi-lemma and uni-lemma modes. In multi-lemma mode, a predicate can have multiple predicate lemmas (e.g., 'run' example in the first paragraph), whereas a predicate can have only one predicate lemma in uni-lemma mode. Languages such as English and Hindi are expected to run in multi-lemma mode using a Document Type Definition (DTD) file, `frameset.dtd`, and other languages such as Arabic and Chinese are expected to run in uni-lemma mode using `verb.dtd`. Although there are two different modes, the interfaces are very similar, so learning one mode effectively teaches the other.

[1] The language-dependent distinction between roleset and frameset is elaborated in Section 4.1

[2] The current version of Cornerstone also runs on JDK 5.0 but running on JDK 6.0 is preferable for the future updates.

2 Getting started

2.1 Install JDK

You first need to install JDK 6.0 or higher on your machine. To install JDK, you need to download the installation file from `http://java.sun.com/javase/downloads/`, and follow the guideline provided by the webpage.

2.2 Download and install Cornerstone

You can download Cornerstone from `http://code.google.com/p/Propbank/downloads/`. From the list, download both `system.tar.gz` and `cornerstone-version.jar` in the same directory. Move to the directory and unarchive `system.tar.gz` by typing the following command on the terminal.

```
tar -zxvf system.tar.gz
```

This command will create two folders: `sys` and `config`. `sys` contains system files required to run Cornerstone. The followings show common systems files used across languages.

```
LANG     : ar (Arabic) | ch (Chinese) | en (English) | hi (Hindi)
LANG.xml : XML template for LANG
LANG.n   : main tags for arguments in LANG (e.g., [0..5], m)
LANG.f   : function tags for modifiers in LANG (e.g., loc, tmp)
```

There are some other system files (e.g., `en.tense`, `ch.src`) that are language specific. The contents of the system files are explained in the following sections. All system files are in text format, so you could update them as needed.

The `config` directory is initially empty, but will be filled with configuration files named after user IDs. Each configuration file (e.g. `choijd`) contains a directory path that indicates the last working directory of the user (see Section 2.3 for more details).

2.3 Launch Cornerstone

Assuming you have downloaded and installed all necessary files, you can launch Cornerstone by typing the following command on the terminal.

```
java -jar cornerstone.jar <language> <user ID>
```

`<language>` can be either `ar`, `ch`, `en` or `hi`, and `<user ID>` is the user ID you want to specify for the tool. For example, if a user `choijd` wants to run Cornerstone in English mode, one needs to type

```
java -jar cornerstone.jar en choijd
```

When you launch Cornerstone, you will see a blank window. Click `[File - Open]` on the menu, move to a directory containing frameset files and choose any frameset file. Cornerstone now shows the contents of the frameset file and creates a configuration file named after `<user ID>` in `config` directory indicating the path of the directory. This configuration file is later used to decide the default directory when you open/save the next frameset file, and is updated whenever you open/save a new frameset file.

3 Cornerstone in multi-lemma mode

3.1 Overview of multi-lemma frameset

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

To open an existing frameset file, click `[File - Open]` on the menu (`Ctrl+O`), move to a directory containing frameset files and choose a frameset file you want to open by scrolling through the list. You could also type the predicate lemma of the frameset you wish to open into the `Filter` box (if it exists) with a `*` before and after the predicate lemma (e.g., `*run*`) in order to restrict the list of file choices. Fig. 1 shows what it looks when you open the frameset file, `run.xml`.

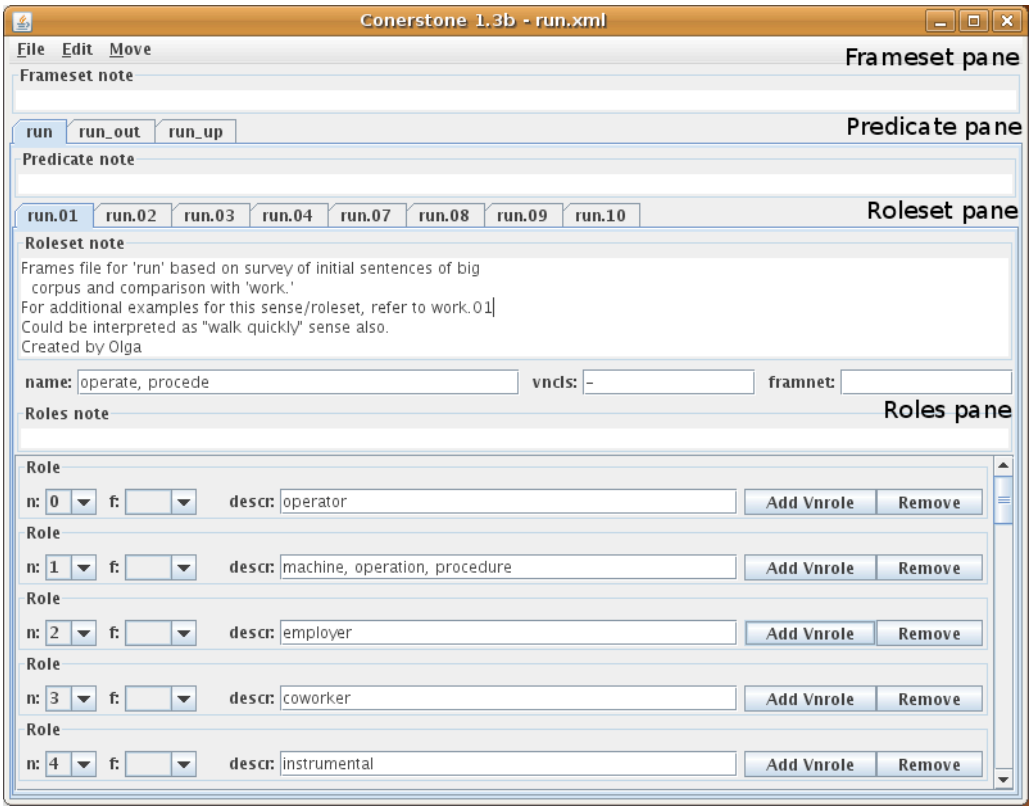

Figure 1: Open `run.xml` frameset file

Multi-predicate mode consists of four panes: frameset pane, predicate pane, roleset pane and roles pane. The frameset pane contains a frameset note, which should be reserved for the relatively unusual occurrence of information that pertains to all predicates and rolesets within the frameset. Additionally, the frameset pane contains the predicate pane.

The predicate pane contains one or more predicate tabs titled by predicate lemmas that may include verb particle constructions (e.g., 'run', 'run out', 'run up'). Each predicate tab contains a predicate note for optional information that pertains to all rolesets encompassed by that predicate. Additionally, each tab contains a roleset pane.

The roleset pane contains several roleset tabs titled by roleset IDs (e.g., `run.01`, `run.02`) for the currently selected predicate (e.g., 'run'). Each roleset tab contains a roleset note for required information about that roleset. This information includes, but is not limited to, the corpus that is the source of that roleset, the VerbNet class 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 (Collin F. Baker, 1998).[3] In addition, the roleset pane contains three attribute fields (`name`, `vncls`, and `framnet`) and a roles pane. The `name` attribute shows

[3]VerbNet and FrameNet information may not be provided in languages other than English.

a brief definition of the current roleset. The `vncls` shows which VerbNet class this roleset is associated with, and `framnet` shows which FrameNet class this roleset is associated with.

The roles pane contains a roles note for optional information about the roles. This may include information that will help annotators disambiguate between roles and may also include syntactic information relevant to the roles. The roles pane also includes one or more roles, which represent arguments that the predicate requires or commonly takes in actual 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 a '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 and semantic roles are intended to be somewhat flexible and can be changed across different predicates; thus each roleset has its own unique argument structure. However, arguments generally correspond to the following semantic roles (Table 1).

`ARG0`	agent	`ARG3`	starting point, benefactive, attribute
`ARG1`	patient	`ARG4`	ending point
`ARG2`	instrument, benefactive, attribute	`ARGM`	modifier

Table 1: List of arguments in Propbank

The function tag, `f` is available for each role, but is not used often because the argument structure outlined in the roles pane includes only high-frequency arguments, which are generally numbered arguments. However, if the survey of a given predicate shows that a certain type of modifier, such as a locative or temporal modifier, is commonly used with the predicate, then the author of the frame can use this tag in place of a numbered argument to add a role labeled `ARGM` (standing for 'modifier') and the appropriate function tag (e.g., `loc`, `tmp`; see Table 2 in page 8 for the complete list of function tags). The attribute field `descr` contains a description of the semantic role, which should be general enough so it can be clearly 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 for `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 that is compatible with the VerbNet information. 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 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 51.3.2. The VerbNet mapping provided by Cornerstone uses only the class number omitting the name. VerbNet also includes thematic roles that should be applicable to all members of a particular class. 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' (see Table 3 for a complete list of VerbNet thematic roles). As mentioned in Section 1, certain predicates may have rolesets that arise in Propbank, but are not compatible with any VerbNet class given for that predicate. Similarly, the Propbank argument structure may include individual roles that are not found in the VerbNet thematic roles, or vice-versa. In these cases, the mappings between the incompatible roleset or role and the VerbNet class or thematic role should be omitted.

precisa já ter o XML, certo?

3.2 Create a new frameset file

To create a new frameset file, click `[File - New]` on the menu (`Ctrl+N`) and type a frameset filename you want to create. This filename should be the main predicate lemma with or without the XML extension (`.xml`; the extension will be added automatically if it is not specified at this point). Figure 2 shows what it looks when you create a frameset file `temp.xml`.

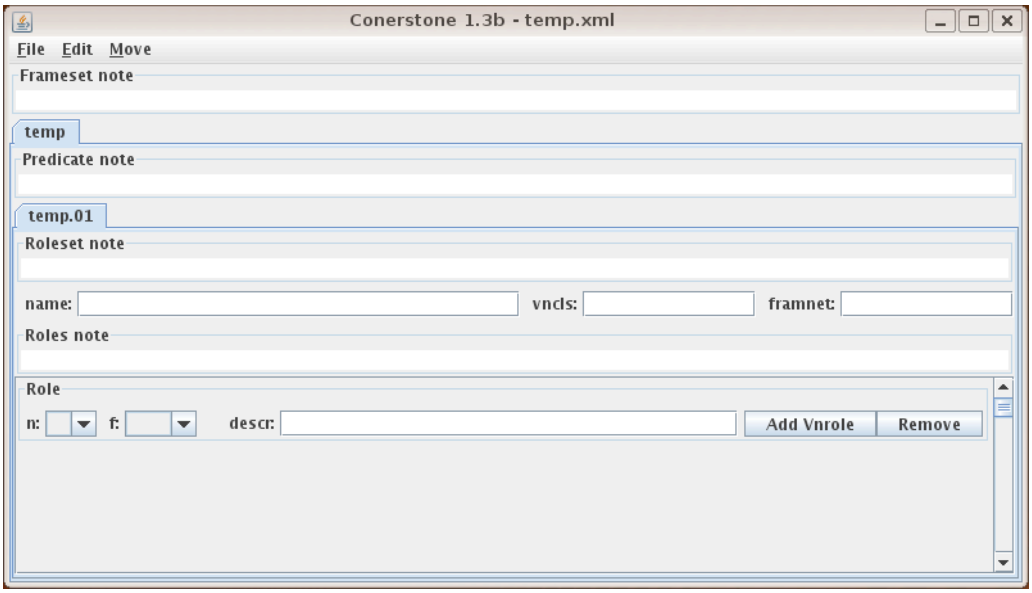

Figure 2: Create `temp.xml` frameset file

When you create a frameset file `temp.xml`, it generates a predicate 'temp' and its roleset `temp.01` by default. To add a new predicate lemma, click `[Edit - Add Predicate]` on the menu (`Ctrl+P`) and type a new predicate lemma. Cornerstone prevents users from generating a new predicate lemma that has the same lemma as any of the existing predicates. For example, you will not be able to add a predicate that has a lemma 'temp' as long as a predicate with the same lemma already exists. Moreover, if the lemma you added includes white spaces, it automatically converts them to underscores ('_'), so the lemma does not include any white space. For example, if you type 'temp in' as a predicate lemma, Cornerstone will convert it to 'temp_in'. As a new predicate is added, Cornerstone also adds its roleset with the following roleset ID (e.g., `temp.02` is added for 'temp_in' and `temp.03` is added for 'temp_out').

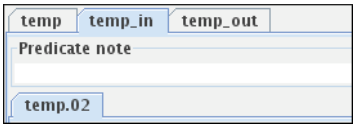

Figure 3: Add predicate lemma 'temp in' and 'temp out'

To change the lemma of the currently selected predicate, click `[Edit - Edit Predicate Lemma]` on the menu (`Ctrl+Alt+P`) and type the new lemma. If you alter the original predicate lemma on the first predicate tab, the roleset IDs will automatically change to reflect the updated predicate lemma. For example, if you alter the predicate 'temp' to 'temporary,' all rolesets IDs will automatically change to `temporary.01`, `temporary.02`, etc. To remove the currently selected predicate, click `[Edit - Remove Predicate]` on the menu (`Ctrl+Shift+P`).

To add a new roleset, click `[Edit - Add Roleset]` on the menu (`Ctrl+R`). Cornerstone automatically generates a roleset ID for the new roleset (e.g., `temp.04`) so you do not need to keep track of the last roleset ID you used. However, the deletion of one predicate leads to the deletion of one or more roleset IDs (e.g., deleting 'temp_in' also deletes `temp.02`), so the addition of the new roleset ID, `temp.04`, could create gaps between roleset IDs. In such situations, the selected roleset can be edited by clicking `[Edit - Edit Roleset ID]` on the menu bar (`Ctrl+Alt+R`).[4] To remove the currently selected roleset, click

[4]This option is not encouraged, unless you try to use an existing frameset file as a template to create a new frameset file.

`[Edit - Remove Roleset]` on the menu (`Ctrl+Shift+R`).

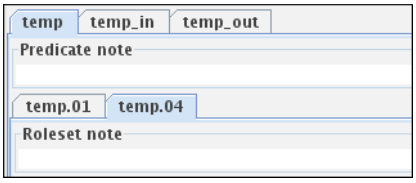

Figure 4: Add a roleset `temp.04` to a predicate 'temp'

To add a role, click `[Edit - Add Role]` on the menu (`Ctrl+L`). You can edit `n` and `f` attributes by clicking the corresponding combo-boxes. When you click `n` combo-box, it gives you a list of argument numbers such as `[0..5]`, `m` for modifiers and `a` for secondary agent. When you click `f` combo-box, it gives you a list of function tags (Table 2).[5] For each role, click within the `descr` field to add a generalized description of the role. To remove a role, click `[remove]` button on the right (Fig. 5).

Tag	Description	Tag	Description
`adv`	adverbial modification	`mod`	modal
`cau`	cause	`neg`	negation
`dir`	direction	`prd`	secondary predication
`dis`	discourse	`prp`	purpose
`dsp`	direct speech	`pnc`	purpose not cause
`ext`	extent	`rcl`	relative clause link
`gol`	goal	`rec`	recipricol (eg herself, etc)
`loc`	location	`slc`	selectional constraint link
`mnr`	manner	`tmp`	temporal

Table 2: List of function tags in English

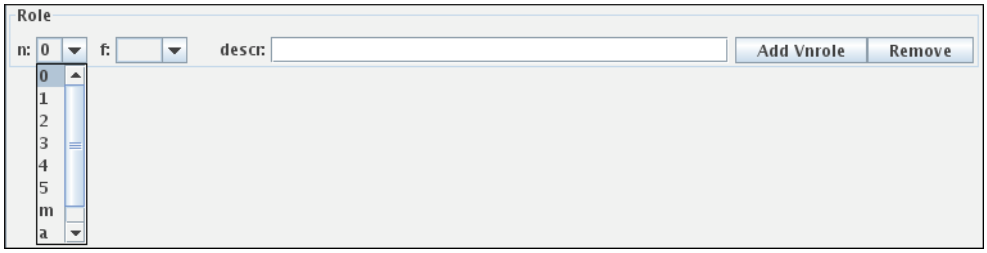

Figure 5: Edit and remove role

[5]English frameset files used to take prepositions as function tags for the arguments of some predicates. Thus, when viewing older frameset files, certain prepositions are listed as function tags, but these are no longer options when you create a new frameset file in Cornerstone.

To add a `vnrole` (VerbNet role), click `[Add Vnrole]` button. For each role, you can add more than one `vnrole`. All VerbNet roles that are appropriate to the currently selected roleset should be included. When you click `vntheta` combo-box, it gives you a list of VerbNet thematic roles (Table 3). To remove a `vnrole`, click `[remove]` button on the right (Fig. 6).

Vntheta	Description
actor1,2	pseudo-agents, used for some communication classes
agent	animate subject, volitional or internally controlled
asset	currency, used for Build/Get/Obtain Classes
attribute	changed quality of patient/theme
beneficiary	entity benefitting from action
cause	entity causing an action, used for psychological/body verbs
destination	end point/target of motion
experiencer	participant that is aware of experiencing something
extent	range or degree of change
instrument	objects/forces that come into contact and cause change in another object
location	underspecified destination/source/place
material	starting point of transformation
patient1,2	affected participants, used for some combining/attaching verbs
predicate	predicative complement
product	end result of transformation
recipient	target of transfer
source	spatial location, starting point
stimulus	events/objects that elicit a response from an experiencer
theme	participants in/undergoing a change of location
theme1,2	indistinct themes, used for differ/exchange classes
patient	affected participants undergoing a process
time	class-specific, express temporal relations
topic	topic of conversation, message, used for communication verbs
proposition	complement clause indicating desired/requested action, used for order class

Table 3: List of VerbNet thematic roles

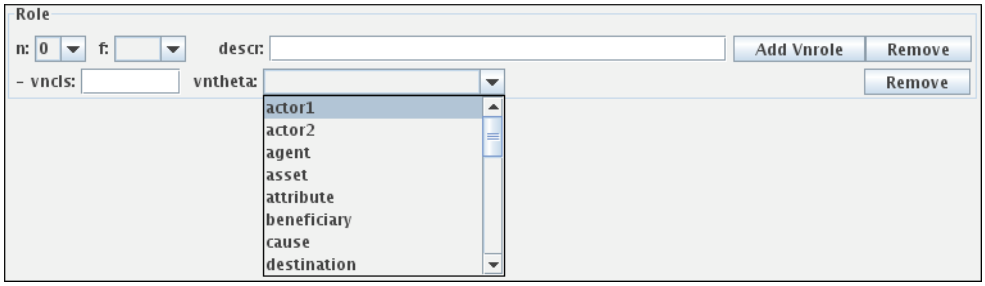

Figure 6: Edit and remove `vnrole` (VerbNet role)

3.3 Edit examples

To view annotated examples of the currently selected roleset, click `[Edit - Edit Examples]` on the menu (`Ctrl+E`), which will prompt a new window: the example frame. Figure 7 shows what it looks when you view examples of a roleset `run.02`. The example frame contains several example tabs titled by the example indices. To add a new example, click `[Edit - Add Example]` on the menu (`Ctrl+=`). To remove the currently selected example, click `[Edit - Remove Example]` on the menu (`Ctrl+−`).

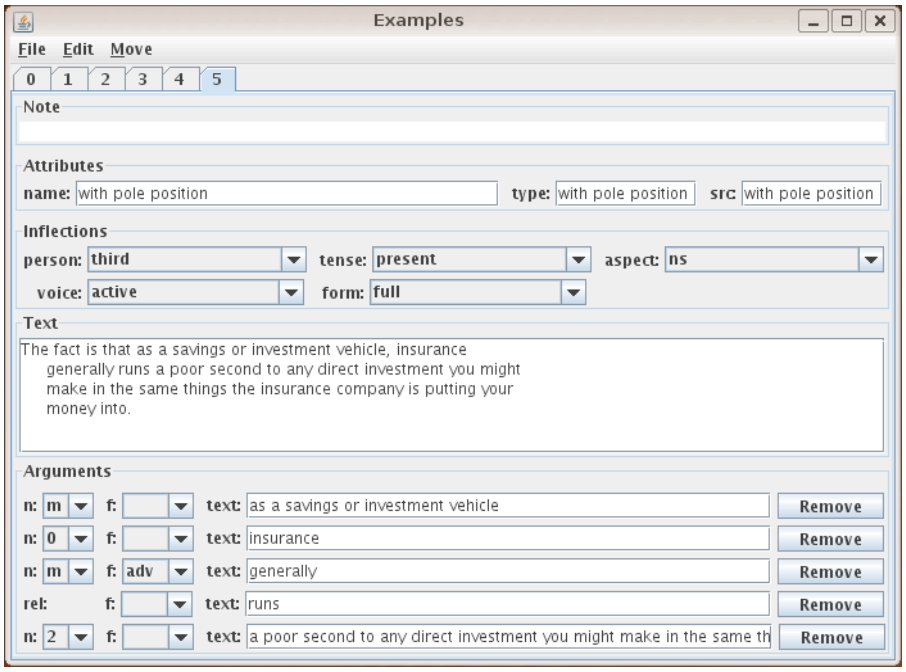

Figure 7: Example frame for `run.02`

Each example tab contains a note space for optional description and helpful information about the current example. In addition, an example tab consists of four panes: attribute pane, inflection pane, text pane, and argument pane. The attribute pane consists of three attribute fields: the `name` field gives a brief identifier of the example, the `type` field can provide optional information about phrasal particle variants and the `src` field shows the type and the title of the corpus that was the source of the example. The inflection pane can be used to provide inflectional information about the example and consists of five attribute fields: `person`, `tense`, `aspect`, `voice`, and `form`. By default, a value *ns* is chosen for each attribute. When you click the inflection combo-box, it gives you a list of attributes shown in Table 4.

Inflection	**Fields**				
`person`	third	other			*ns*
`tense`	present	past	future		*ns*
`aspect`	perfect	progressive	both		*ns*
`voice`	active	passive			*ns*
form	infinitive	gerund	participle	full	*ns*

Table 4: List of inflections

The text pane contains the actual example in text. The text examples should include the relevant indices between traces and explicit mentions so that readers can fully understand the syntax of the example. This may require bracketing an indexed constituent in order to convey what the constituent boundaries are. For example, the passive sentence, "The car on the left was hit by the minivan" should be expressed as "`[` The car on the left `]-1` was hit `*T*-1` by the minivan".

The argument pane contains a set of arguments and the relation (or predicate). Each argument consists of three attribute fields: `n` is an argument number, `f` is a function tag, and `text` contains the portion of the example that constitutes the given argument. Similarly, each relation consists of two attribute fields: `f` and `text`; however, only the `text` field is necessary to indicate the predicate. To add an argument, choose `[Edit - Add Argument]` on the menu (`Ctrl+9`). To remove an argument/relation, click `[Remove]`

button on the right. To add the relation, choose `[Edit - Add Relation]` on the menu (`Ctrl+0`). If the relation includes a verb particle construction, the individual words should be bracketed to indicate the concatenation of two words into a single relation (e.g., rel: `[run][up]`). Examples are automatically saved when the example window is closed. To close the example window click `[File - Quit]` on the menu (`Ctrl+W`).

Certain arguments that include relative clause and/or reduced relative links require special notation in the `text` field. For relative clauses, the relative pronoun (or placeholder `0` for gapped pronouns) with its index is followed by `*-->` and the referent that the relativizer is linked to. For example, the sentence,

`[The man]-1 that-1 *PRO*-1` walked

will have an argument with `n:m`, `f:SLC` and `text: that-1 *--> The man` to indicate that the correct annotation includes a manually added `*` link between the relative pronoun and its referent. For reduced relative constructions, the trace argument is followed by `&-->` and the referent of the trace. For example, the sentence,

The man killed `*none*` was my friend

where the relation is 'kill', will have an argument with `n:1`, `text: *none* &--> The man` to indicate that the annotation includes a manually added `&` link between the trace and its referent.

Each roleset should be accompanied by at least one example. Ideally, example(s) should be drawn directly from the corpus that the roleset arose in to reflect accurate usage. However, in some cases, the examples that arise in the corpus may not give the annotators a helpful illustration of the predicate's argument structure, or may not include examples of potentially confusing additional arguments. While at least one example from the corpus should always be included, the author of the roleset may add additional examples (either invented or through an Internet search) so that the predicate's argument structure is fully illustrated and understandable to the annotator.

3.4 Save a frameset file

Cornerstone automatically saves your previous work as you open or create a new frameset file. To save a frameset file manually, click `[File - Save]` on the menu (`Ctrl+S`). To save the frameset file as a different file, click `[File - Save As]` on the menu (`Ctrl+Shift+S`) and type the new frameset filename.

4 Cornerstone in uni-lemma mode

4.1 Overview of uni-lemma frameset

Languages such as Arabic and Chinese are expected to run in *uni-lemma* mode. Unlike multi-lemma mode that allows a verb to have multiple predicate lemmas (e.g., 'run', 'run out', 'run up'), uni-lemma mode allows only one predicate lemma for a verb. The XML structure of the uni-lemma frameset file is defined in a DTD file, `verb.dtd`.

To open an existing frameset file using Cornerstone, click `[File - Open]` on the menu (`Ctrl+` O), move to a directory containing frameset files and choose a frameset file you want to open. Fig. 8 shows what it looks when you open an Arabic frameset file `HAfaZ.xml`.

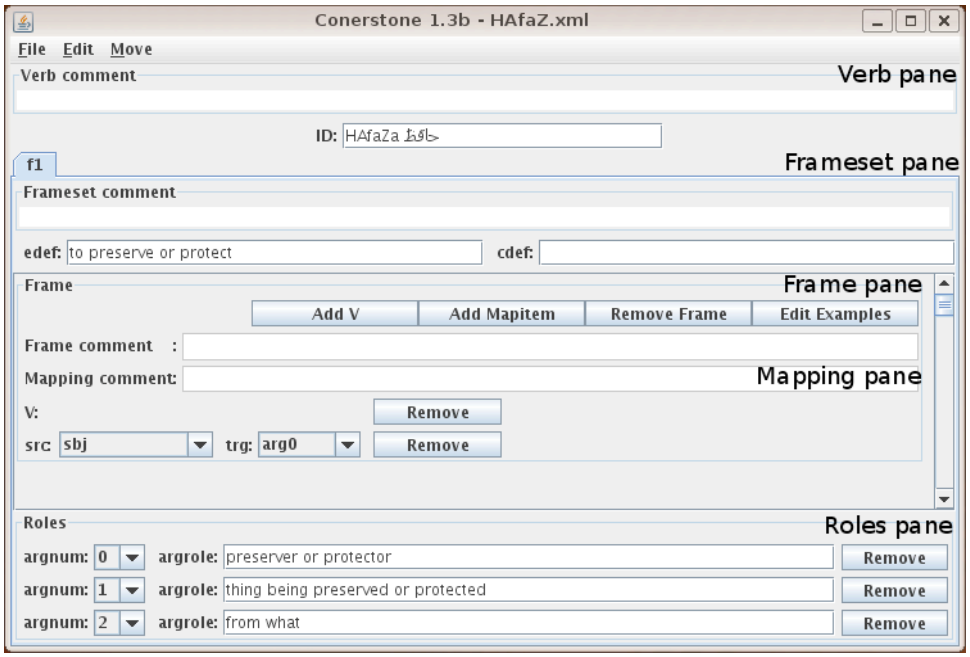

Figure 8: Open `HAfaZ.xml` frameset file

Uni-lemma mode consists of four panes: 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` for the predicate lemma of the verb (a lemma can be represented either in Roman alphabets or characters in other languages). Additionally, the verb pane contains the frameset pane as its sub-pane.

The frameset pane contains several frameset tabs titled by frameset IDs for the verb. Note that the frameset in uni-lemma mode is equivalent to the roleset in multi-lemma mode. Unlike multi-lemma mode in which roleset IDs are assigned with the main predicate lemma followed by sequential numbers (e.g., `run.01`, `run.02`), frameset IDs in uni-lemma mode are assigned with `f` (standing for 'frameset') followed by sequential numbers (e.g., `f1`, `f2`). The frameset pane also contains a frameset comment for required information about the currently selected frameset and two attribute fields, `edef` and `cdef`, that 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. The mapping pane contains a mapping comment that describes mappings between syntactic and semantic arguements associated with the frameset. It also contains `V` (standing for 'verb') and a set of mappings. `V` is a placeholder indicating where the verb predicate should be located among the other arguments. The mapping shows an association between a syntactic argument, `src` (e.g., subject, object) and a semantic argument, `trg` (e.g., agent, patient) of the frameset. The syntactic arguments are sometimes provided in the Treebank, in which case, these mappings can be used for automatic extraction of semantic arguments from their syntactic arguments. Table 5 (page 14) shows the full list of syntactic arguments.

The roles pane consists of a set of arguments that the predicate requires or commonly takes in actual usage. Each argument has two attributes fields: `argnum` is an argument number and `argrole` shows a description of the semantic role. The list of Propbank arguments is provided in Table 1 (page 6).

4.2 Create a new frameset file

To create a new frameset file, click [File - New] on the menu (Ctrl+ N) and type a frameset filename you want to create. The filename may or may not include XML extension (.xml), which will be added automatically if it is not specified. Fig. 9 shows what it looks when you create a frameset temp.xml.

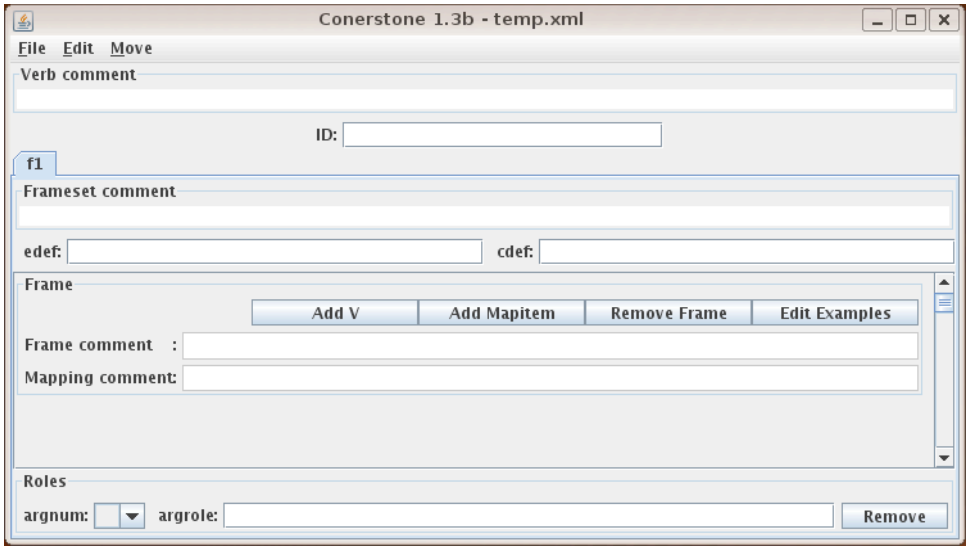

Figure 9: Create temp.xml frameset file

When you create a new frameset file, it generates a frameset f1 by default. To add a new frameset, click [Edit - Add Frameset] on the menu (Ctrl+ F). Cornerstone automatically generates a frameset ID for the new frameset (e.g., f2) so you do not need to keep track of the last frameset ID you used. To remove the currently selected frameset, click [Edit - Remove Frameset] on the menu (Ctrl+Shift+F).

Figure 10: Add frameset f2 and f3 to temp.xml

To add a new frame, click [Edit - Add Frame] on the menu (Ctrl+R). To remove the frame, click [Remove Frame] button on the frame (Fig. 11).

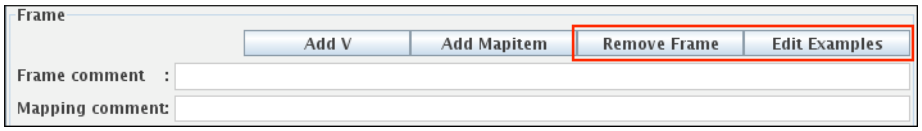

Figure 11: Add Remove a frame

To add a verb-placeholder or a mapping to the frame, click [Add V] or [Add Mapitem] button, respectively. You can add more than one mapping for each frame. When you click src combo-box, it gives you a list of syntactic arguments (Table 5). Similarly, when you click trg combo-box, it gives you a list of semantic arguments (Table 1, page 6). To remove a verb-placeholder or a mapping, click [Remove] button on the right (Fig. 12).

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

Table 5: List of syntactic sources

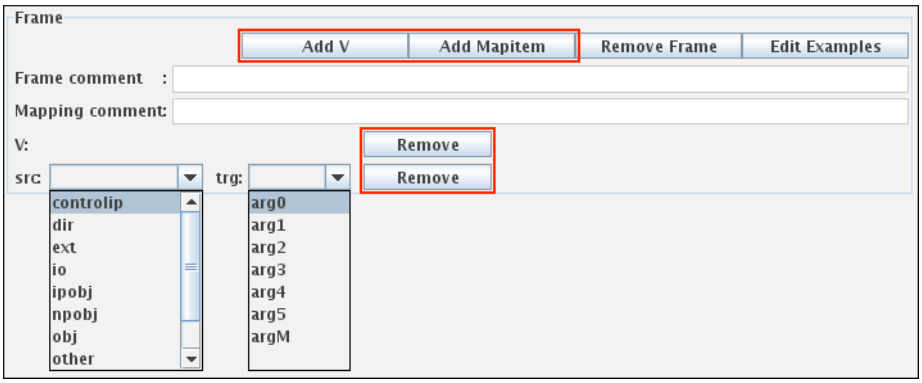

Figure 12: Add, edit and remove mappings

To add an argument, click `[Add Role]` on the menu (`Ctrl+` L). When you click `argnum` combo-box, it gives you a list of argument numbers such as `[0..5]` and `m` for modifiers. For each argument, click within `argrole` field to add a generalized description of the argument. To remove an argument, click `[Remove]` button on the right (Fig. 13).

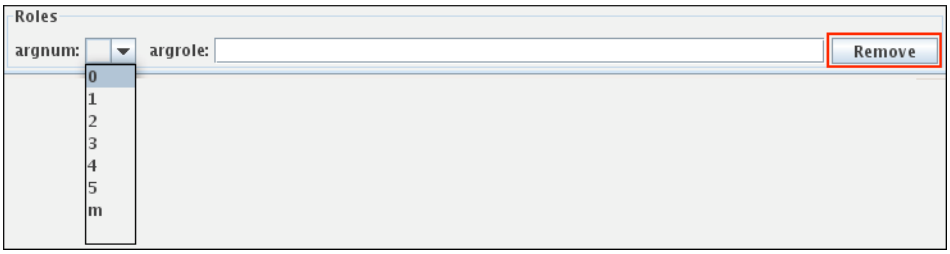

Figure 13: Edit and remove an argument

4.3 Edit examples

To view examples of the frame, click the `[Edit Examples]` button (Fig. 11, page 13), which will prompt a new window: the example frame. Fig. 14 shows how it looks when you view examples of a frameset `f2` in `HAfaZ.xml`. The example frame contains several example tabs titled by the example indices (starting with 0). To add a new example, click `[Edit - Add Example]` on the menu (`Ctrl+=`). To remove the currently selected example, click `[Edit - Remove Example]` on the menu (`Ctrl+−`).

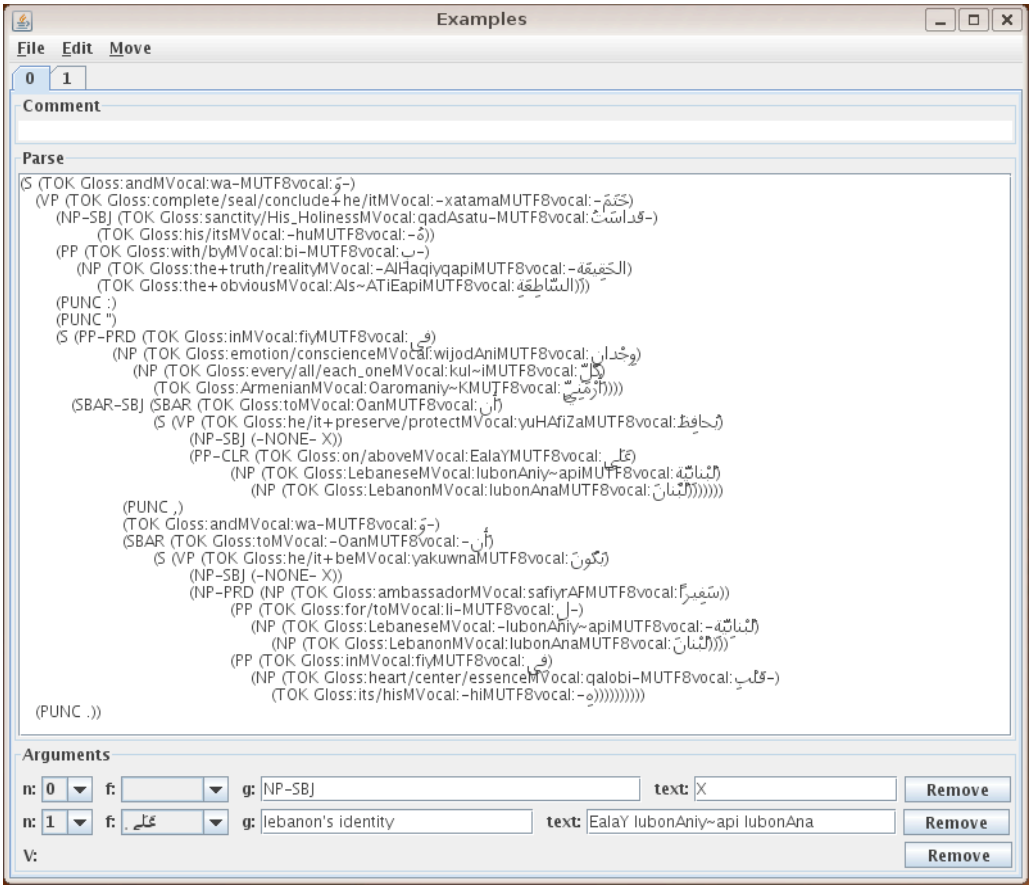

Figure 14: Example frame for `f2` in `HAfaZ.xml`

Each example tab contains a comment space for optional description about the current example. In addition, an example tab consists of two panes: parse pane and argument pane. Unlike the text pane in multi-lemma mode (Section 3.3), the parse pane contains the actual example in Treebank format.

The argument pane contains a verb-placeholder and a set of arguments. `V` is a placeholder indicating where the verb should be located among the other arguments. Each argument consists of three (or four) attribute fields: `n` is an argument number, `f` is a function tag, `text` shows the portion of the example that constitutes the given argument and `g` shows the English translation of `text` filed (`g` field currently exists only in Arabic frameset files). To add an verb-placeholder/argument, click `[Edit - Add Verb/Argument]` on the menu (`Ctrl+0/9`). To remove a verb-placeholder/argument, click `[Remove]` button on the right.

4.4 Save a frameset file

Cornerstone automatically saves your previous work as you open or create a new frameset file. To save a frameset file, click `[File - Save]` on the menu (`Ctrl+S`). To save the frameset file as a different file, click `[File - Save As]` on the menu (`Ctrl+Shift+S`) and type the new frameset filename.

5 Summary

For each verb predicate in Propbank, there exists a corresponding frameset file that contains information about the senses associated with the verb, and also defines each sense's predicate argument structure. Since most Propbank annotations are based on the frameset files, it is important to keep the files consistant as well as easy to update. Currently, all frameset files are written in XML format, which is difficult to edit using a plain text editor. This motivated the development of a new tool dedicated to the creation of frameset files, Cornerstone.

By using Cornerstone, you can view, create and edit frameset files without having any knowledge of XML. Furthermore, the frameset files created using Cornerstone are guaranteed to be free of the errors that commonly occur when directly manipulating XML. Cornerstone can be run in two modes, multi-lemma mode and uni-lemma mode. Multi-lemma mode allows each verb to have multiple predicate lemmas, whereas uni-lemma mode allows only one predicate lemma for a verb. While the two different modes ensure flexible use of Cornerstone with a variety of different languages, the interfaces are very similar, so learning one mode effectively teaches the other.

Developing a new tool is always challenging; yet, it is worthwhile because once we have a tool that meets our needs, the annotation process can be made faster and more accurate. We will continue to develop the tool by improving its functionalities through user-testing and feedback, and by applying it to more languages.

References

John B. Lowe Collin F. Baker, Charles J. Fillmore. 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 (COLING-ACL'98)*.

Mona Diab, Aous Mansouri, Martha Palmer, Olga Babko-Malaya, Wajdi Zaghouani, Ann Bies, and Mohammed Maamouri. 2008. A pilot arabic propbank. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'08)*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*, volume Language, Speech and Communications. MIT Press.

Chunghye Han, Narae Han, Eonsuk Ko, and Martha Palmer. 2002. Korean treebank: Development and evaluation. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC'02)*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending verbnet with novel verb classes. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06)*.

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.