PropBank Annotation Guidelines

Claire Bonial
bonial@colorado.edu

Olga Babko-Malaya
malayao@ldc.upenn.edu

Jinho D. Choi
choijd@colorado.edu

Jena Hwang
hwangd@colorado.edu

Martha Palmer
mpalmer@colorado.edu

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

Annotation Guidelines (Version 3.0)

January 24, 2011

Contents

Chapter 1

Verb Annotation Instructions

1.1 PropBank Annotation Goals

PropBank is a corpus in which the arguments of each predicate are annotated with their semantic roles in relation to the predicate (Palmer et al., 2005). Currently, all the PropBank annotations are done on top of the phrase structure annotation of the Penn TreeBank (Marcus et al., 1993). In addition to semantic role annotation, PropBank annotation requires the choice of a sense ID (also known as a 'frameset' or 'roleset' ID) for each predicate. Thus, for each verb in every tree (representing the phrase structure of the corresponding sentence), we create a PropBank instance that consists of the sense ID of the predicate (e.g., `run.02`) and its arguments labeled with semantic roles.

An important goal is to provide consistent argument labels across different syntactic realizations of the same verb, as in. . .

```
[ARG0 John] broke [ARG1 the window]
[ARG1 The window] broke
```

As this example shows, the arguments of the verbs are labeled as numbered arguments: Arg0, Arg1, Arg2 and so on. The argument structure of each predicate is outlined in the PropBank frame file for that predicate. The frame file gives both semantic and syntactic information about each sense of the predicate lemmas that have been encountered thus far in PropBank annotation. The frame file also denotes the correspondences between numbered arguments and semantic roles, as this is somewhat unique for each predicate. Numbered arguments reflect either the arguments that are required for the valency of a predicate (e.g., agent, patient, benefactive), or if not required, those that occur with high-frequency in actual usage. Although numbered arguments correspond to slightly different semantic roles given the usage of each predicate, in general numbered arguments correspond to the following semantic roles:

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

Table 1.1: List of arguments in PropBank

In addition to numbered arguments, another task of PropBank annotation involves assigning functional tags to all modifiers of the verb, such as manner (MNR), locative (LOC), temporal (TMP) and others:

Mr. Bush met him privately, in the White House, on Thursday.
Rel: met

Arg0: Mr. Bush
Arg1: him
ArgM-MNR: privately
ArgM-LOC: in the White House
ArgM-TMP: on Thursday

And, finally, PropBank annotation involves finding antecedents for empty arguments of the verbs, as illustrated below:

*I made a decision [*PRO*] to leave.*

The subject of the verb leave in this example is represented as an empty category [*] in TreeBank. In PropBank, all empty categories which could be co-referred with a NP within the same sentence are linked in co-reference chains:

Rel: leave
*Arg0: [*PRO*] * [I]*

Similarly, relativizers and their referents are linked in relative clause constructions, and traces are linked to their referents in reduced relative constructions.

The annotation of this information creates a valuable corpus, which can be used as training data for a variety of natural language processing applications. Training data, essentially, is what computer scientists and computational linguists can use to 'teach the computer' about different aspects of human language. Once this information is processed, it can guide future decisions on how to categorize and/or label different features in novel utterances outside of the PropBank corpus. Parallel PropBank corpora currently exist or are underway for English, Chinese, Arabic and Hindi. As a whole, the PropBank corpus has the potential to assist in natural language processing applications such as machine translation, text editing, text summary and evaluation as well as question answering.

Thus, the main tasks of PropBank annotation are: argument labeling, annotation of modifiers, choosing a sense for the predicate, and creating links for empty categories, relative clauses, and reduced relatives. Each of these aspects of annotation are discussed in detail below. Although some detail is provided in each section on how to annotate appropriately using the annotation tool, Jubilee, complete guidelines on the use of this tool are provided in chapter 4.

1.2 Sense Annotation

1.2.1 Frame Files

The argument labels for each verb are specified in the frame files, which are available at http://verbs.colorado.edu/propbank/framesets-english/ and are also displayed in the frameset view of the annotation tool, Jubilee (see 4 for further information). Frame files provide verb-specific descriptions of semantic roles and illustrate these roles by providing examples.

Frame File for the verb 'expect':
Roles:
Arg0: expecter
Arg1: thing expected

Example
Portfolio managers expect further declines in interest rates.

Arg0: Portfolio managers
REL: expect
Arg1: further declines in interest rates

For some verbs, it is impossible to provide one set of semantic roles for all senses of the verb. For example, the two senses of the verb 'leave' in the examples below take different arguments:

Mary left the room
Mary left her daughter-in-law her pearls in her will

In such cases, frame files distinguish two or more verb senses, which are called framesets or rolesets (this term is interchangeable, depending on what language is being annotated), and define argument labels specific to each frameset:

Roleset leave.01 'move away from':
Arg0: entity leaving
Arg1: place left

Roleset leave.02 'give':
Arg0: giver
Arg1: thing given
Arg2: beneficiary

As previously mentioned, frame files are also found in the frameset view (upper-right pane) of the annotation tool, Jubilee. Initially, the predicate lemma followed by `.XX` is displayed here to indicate that no sense has been chosen yet (e.g., `leave.XX`).

When annotating, annotators first select the appropriate roleset (or sense), and then assign the argument labels as specified for this roleset. In Jubilee, annotators can peruse the available numbered senses of a predicate by clicking on the roleset combo-box, or they can move through the available rolesets sequentially by using the shortcut `]` to move forward, or `[` to move back to lower-numbered senses. As a roleset is selected, the argument structure and a short definition of that sense, which are extracted from the corresponding frameset file (e.g., `leave.xml`), appear in the roleset information pane. To view annotation examples of the currently selected roleset, click `[Example]` button (`Ctrl`+E).

To accommodate verb particle constructions (e.g. give up) , the frame file defines not only several senses of each verb, but also several predicates reflective of the verb's associated verb particle constructions. If a verb has a particle (marked as PRT in TreeBank), then it is considered a different predicate, and may or may not have a different set of semantic roles. For example, the frame file for the verb 'keep' defines three predicates: predicate 'keep' (which has 6 rolesets), and predicates 'keep up' and 'keep on,' which each encompass 1 roleset respectively. The following example gives the definition of the predicate 'keep up' and an example usage:

Predicate: keep up:
keep.05 'keep up: maintain position':
Arg0: maintainer of position
Arg1: relative to what

John can't keep up with Mary's rapid mood swings.
Arg0: John

Argm-MOD: ca
Argm-NEG: n't
REL: keep up
Arg1: with Mary's rapid mood swings

Note that the relation (REL) in PB annotation should include both the verb and the particle. Thus, the annotator must concatenate the particle to the original relation to form a single predicate lemma, annotated with the '`rel`' tag. To concatenate the particle, choose the particle node (as opposed to selecting just the particle itself) and type `Ctrl+Shift+/`. The resulting `rel` annotation will reflect the locations of both the original predicate and the concatenated particle in the annotation view; for example, `9:0,8:0-rel`.

1.2.2 ER: Error roleset

All tokens marked as verbs in the TreeBank should be annotated in PropBank; however, rarely a token is marked as the REL that is not truly a verb and should not be annotated. Because the lines between parts of speech are often fuzzy, annotators should annotate all cases of gerunds and past participles, even if they seem adjectival or nominal in usage. However, if the token marked as a verb is not ever used grammatically as an active verb, then it should not be annotated and the ER roleset should be selected. For example, 'collonaded' in *The collonaded house...* has been marked as a verb in the past in the TreeBank, but a web search shows that there are no attested usages of 'collonade' as an active verb; thus, this instance was treated as an error and marked as ER. This roleset should also be selected when a verb is being used prepositionally, and therefore heads a prepositional phrase in the TreeBank. For example, although 'accord' can appear as an active verb, prepositional usages such as *According to our sources...* should be marked as ER. Other examples of verbs that are often used prepositionally are 'base' and 'give' in usages such as *Based on current research...* and *Given the situation...* In each of these cases, the annotator will notice that the verb syntactically heads a prepositional phrase. In general, ER should be selected for error cases where the REL is not a verb.

1.2.3 IE: Idiomatic Expression roleset

If the REL is part of an idiomatic expression, wherein the meaning of the expression has no relationship to the meaning of its parts, then the IE roleset should be selected. Because PropBank has coarse-grained senses which treat metaphorical extensions of a sense in the same manner as the literal sense, annotators should be careful not to use IE where the idiomatic expression is metaphorically related to the words that comprise it. For example, 'let' in *let the cat out of the bag* should be annotated because 'the cat' is metaphorically related to a secret and 'the bag' is related to the secret's hidden nature. However, 'trip' in *Trip the light fantastic toe* (meaning 'to dance'), has no relationship to dancing, and similarly, 'kick' in *Kick the bucket* has no relationship to dying. In these cases, and only these cases, where the meaning of the expression is in no way related to the individual meanings of the words, the IE roleset should be selected.

1.2.4 DP: Duplicate indicator roleset

This roleset is not available to annotators, but is used in post-processing to indicate that a duplicate was necessary to handle a verb that has two separate argument structures. This occurs only in cases of verb ellipsis, for example, *I ate a sandwich and Cindy a banana.* The TreeBank uses special '=' notation and numbered indices to indicate that the second argument structure shares a verb given earlier. In these cases, annotators should select the appropriate numbered roleset, and should annotate only the first argument structure. Then, the annotator should fill out

a problem report on the PropBank website (http://verbs.colorado.edu/propbank/), indicating that the instance needs a duplicate. During adjudication and post-processing, this duplicate is added and the second argument structure is annotated. The annotation of the second argument structure receives the DP roleset during post-processing so that it is clear that this instance is a duplicate.

1.2.5 LV: Light Verb roleset

This roleset is used to flag a verb's usage as a light verb usage (e.g. *take a walk, have a drink*). See Chapter 2 for more details on light verb annotation.

1.2.6 What to do when there is no frame file

Occasionally, annotators will come across new verbs that do not have existing frame files. In these cases, firstly check to see if this is a mislemmatization of a verb that already has a frame file (i.e., a British or misspelling). Use the frames listed on the website (http://verbs.colorado.edu/propbank/framesets-english/) to check this. If it is a mislemmatization, use the argument structure outlined in the existing frame file and take a note of the task and instance number of this problem, along with the correct roleset (e.g., color.01). Using this information, fill out a problem report on the PropBank website (http://verbs.colorado.edu/propbank/). If this is not the case, the annotator will have to do some research to determine what an appropriate argument structure would be. Brainstorm other verbs that have similar syntax and semantics, and see if any of those verbs already have frame files. Try to model your annotation after that frame file. Again, fill out a problem report for this instance, taking note of the task and instance number along with an outline of the numbered arguments used and their semantic role correspondences. Also, include the roleset that served as a basis of comparison if one was used.

If desired, the annotators can consult the Unified Verb Index: https://verbs.colorado.edu/verb-index/. This has links to existing VerbNet and FrameNet information on a particular predicate, which can be used to understand what are commonly thought of as a verb's core arguments. Where possible, PropBank is mapped to VerbNet thematic roles. If a mapping is appropriate, a roleset's VN class will be listed at the top of the roleset on the frames website. VerbNet uses more canonical thematic roles instead of numbered arguments that are unique to the predicate. Thus, the annotator can use this as a resource for brainstorming argument structures for new verbs as well as coming to a better understanding of existing frame files by examining another analysis of that verb's thematic roles.

1.3 Annotation of Numbered Arguments

1.3.1 Choosing Arg0 versus Arg1

In most cases, choosing an argument label is straightforward, given the verb specific definition of this label in the frame files. However, in some cases, it may be somewhat ambiguous whether an argument should be annotated as Arg0 or Arg1; thus, the annotator must decide between these labels based on the following explanations of what generally characterizes Arg0 and Arg1.

The Arg0 label is assigned to arguments which are understood as agents, causers, or experiencers. The Arg1 label is usually assigned to the patient argument, i.e. the argument which undergoes the change of state or is being affected by the action.

Arg0 arguments (which correspond to external arguments in GB theory) are the subjects of transitive verbs and a class of intransitive verbs called unergatives.

John (Arg0) sang the song.
John (Arg0) sang.

Semantically, external arguments have what Dowty (1991) called Proto-Agent properties, such as

```
-volitional involvement in the event or state
-causing an event or change of state in another participant
-movement relative to the position of another participant
```

(Dowty, 1991)

Internal arguments (labeled as Arg1) are the objects of transitive verbs and the subjects of intransitive verbs called unaccusatives:

John broke the window (Arg1)
The window (Arg1) broke

These arguments have Proto-Patient properties, which means that these arguments

```
-undergo change of state
-are causally affected by another participant
-are stationary relative to movement of another participant
```

(Dowty, 1991)

Whereas for many verbs, the choice between Arg0 or Arg1 does not present any difficulties, there is a class of intransitive verbs (known as verbs of variable behavior), where the argument can be tagged as either Arg0 or Arg1.

A bullet (Arg1) landed at his feet
He(Arg0) landed

Arguments which are interpreted as agents should always be marked as Arg0, independent of whether they are also the ones which undergo the action. In general, if an argument satisfies two roles, the highest ranked argument label should be selected, where

```
Arg0 > Arg1 >  Arg2 >
```

Given this rule, agents are ranked higher than patients. If an argument is both an agent and a patient, then Arg0 label should be selected.

Not all Arg0s are agentive, however. There are many inanimate as well as clausal arguments which are being marked as Arg0s. These arguments are usually the ones which cause an action or a change of state.

A notion which might be useful for selecting Arg0 arguments is the notion of internally caused as opposed to externally caused eventualities, as defined in Levin and Rapapport (1995). In internally-caused eventualities, some property inherent to the argument of the verb is responsible for bringing about the eventuality. For agentive verbs such as *play, speak*, or *work*, the inherent property responsible for the eventuality is the will or volition of the agent who performs the activity. However, an internally caused eventuality need not be agentive. For example, the verbs *blush* and *tremble* are not agentive, but they, nevertheless, can be considered to denote internally caused eventualities, because these eventualities arise from internal properties of the arguments, typically an emotional reaction. In contrast to internally caused verbs, verbs which are externally caused inherently imply the existence of an external cause with an immediate

control over bringing about the eventuality denoted by the verb: an agent, and instrument, a natural force, or a circumstance. Thus something breaks because of the existence of some external cause; something does not break because of its own properties (Levin and Hovav, 1995). The difference between internal and external causation is important for distinguishing Arg0s and Arg1s: the arguments which are responsible for bringing out the eventuality are Arg0s, whereas those which undergo an externally caused event are Arg1s.

To sum up, Arg0 arguments are the arguments which cause the action denoted by the verb, either agentively or not, as well as those which are traditionally classified as experiencers, i.e. the arguments of stative verbs such as love, hate, fear. Arg1 arguments, on the other hand, are those that change due to external causation, as well as other types of patient-like arguments.

1.3.2 Arg-A: Secondary Agent

In addition to argument numbers 1-5 and modifiers, the tag 'Arg-A' is available. This should be used to annotate secondary agents. Verbs that often take secondary agents will have this specified in the roleset and a clarifying example will be provided (e.g., walk.01); however, it is possible albeit rare for other verbs to be characterized by a secondary agent. In general, the secondary agent tag will only be used when the argument structure outlined in the roleset indicates that a proto-agent role, such as 'the walker' - Arg-0 for sense walk.01, is already fulfilled, yet there is another animate agent causing the event:

John walked his dog
ARG-A: John
REL: walked
ARG-0: his dog

1.4 Annotation of Modifiers

The following types of modifiers are being used in PropBank:

COM: Comitative
LOC: Locative
DIR: Directional
GOL: Goal
MNR: Manner
TMP: Temporal
EXT: Extent
REC: Reciprocals
PRD: Secondary Predication
PRP: Purpose
CAU: Cause
DIS: Discourse
ADV: Adverbials
ADJ: Adjectival
MOD: Modal
NEG: Negation
DSP: Direct Speech
SLC: Relative Clause
LVB: Light Verb

Note: In the sections that follow describing each type of modifier, many real examples drawn from the corpus are used. As such, these examples contain null elements (*, PRO) and traces (*T*). These null elements and traces often use indices, or numbers, to show the relationship between a null element and its referent. Thus, the null element or trace may have a number listed after it, and that number is listed again next to a word or phrase. This indicates that the two are coreferential.

1.4.1 Comitatives (COM)

Comitative modifiers indicate who an action was done *with*. This can include people or organizations (entities that have characteristics of prototypical agents: animacy, volition) but excludes objects, which would be considered instrumental modifiers. Although the formal term for this modifier is 'comitative,' annotators can think of this argument as 'companion:' a companion to another in the action of the verb.

I sang a song with my sister.
ARG0: I
REL: sang
ARG1: a song
ARGM-COM: with my sister

The man joined the club with his friend.
ARG0: The man
REL: joined
ARG1: the club
ARGM-COM: with his friend

I-1 got kicked [-1] out of the class with the bully.*
REL: kicked
ARG1: [*-1]
ARGM-DIR: out of the class
ARGM-COM: with the bully

The next morning, with a police escort, busloads of executives and their wives raced to the Indianapolis Motor Speedway.
ARGM-TMP: The next morning
ARGM-COM: with a police escort
ARG0: busloads of executives and their wives
REL: raced
ARG1: to the Indianapolis Motor Speedway

1.4.2 Locatives (LOC)

Locative modifiers indicate where some action takes place. The notion of a locative is not restricted to physical locations, but abstract locations are being marked as LOC as well; such as '[in his speech]-LOC he was talking about. . .'

The percentage of lung cancer deaths among the workers at the West Groton , Mass. , paper factory appears [-1] to be the highest for [any asbestos workers]-1 studied [*-1] in Western industrialized countries , he said [0] [*T*-2] .*

ARG1: [*]
REL: studied
ARGM-LOC: in Western industrialized countries

*Areas of the factory [*ICH*-2] were particularly dusty where-1 [the crocidolite]-8 was used [*-8] [*T*-1] .*
LINK-SLC: where-1 * [Areas of the factory]
ARGM-LOC: [*T*-1]
ARG1: [*-8]
REL: used

*In his ruling , Judge Curry added an additional $ 55 million [*U*] to the commission 's calculations.*
ARGM-LOC: In his ruling
ARG0: Judge Curry
REL: added
ARG1: an additional $ 55 million [*U*]
ARG2: to the commission 's calculations

1.4.3 Directional (DIR)

Directional modifiers show motion along some path. 'Source' modifiers are also included in this category. However, if there is no clear path being followed, a 'location' marker should be used instead. Thus, 'walk along the road' is a directional, but 'walk around the countryside' is a location. Directional modifiers are also used for some particles, as in 'back up'.

*What sector is [*T*-46] stepping forward [*-2] to pick up the slack ? " he asked [*T*-1]*
ARG1: [*T*-46]
REL: stepping
ARGM-DIR: forward
ARGM-PRP: [*-2] to pick up the slack

*That response annoyed Rep. Markey , House aides said [0] [*T*-1] , and the congressman snapped back that there had been enough studies of the issue and that it was time for action on the matter .*
ARG0: the congressman
REL: snapped
ARGM-DIR: back
ARG1: that there had been enough studies of the issue and that it was time for action on the matter

1.4.4 Goal (GOL)

This tag is for the goal of the action of the verb. This includes the final destination of motion verbs and benefactive arguments that receive something, or modifiers that indicate that the action of the verb was done for someone or something, or on their behalf:

The child fed the cat for her mother.
ARG0: the child

RELl: fed
ARG1: the cat
ARGM-GOL: for her mother

The couple translated for the Americans.
ARG0: the couple
REL: translated
ARGM-GOL: for the Americans

'Goal' will also be used for modifiers that indicate the final resting place or destination of motion or transfer verbs:

Workers dumped large burlap sacks of the imported material into a huge bin , poured in cotton and acetate fibers and mechanically mixed the dry fibers in a process used [*] [*] to make filters.
ARG0: Workers
REL: dumped
ARG1: large burlap sacks of the imported material
ARGM-GOL: into a huge bin

We publicized to the masses our determination to fight against evil.
ARG0: We
REL: publicized
ARGM-GOL: to the masses
ARG1: our determination to fight against evil

The walls crumbled to the ground.
ARG1: The walls
REL: crumbled
ARGM-GOL: to the ground

Be careful to distinguish instances like that of the above example from secondary predication (e.g. 'he bled **to death**'); the difference being that the above example involves motion and coming to rest.

Many motion verbs involving a change of state, such as 'rise,' and 'fall,' already have a numbered argument for this semantic role. Similarly, many transfer verbs, such as 'give,' and 'distribute,' already have a numbered argument for this role. In these cases, as in all situations where we have numbered arguments that are also encompassed by argMs, continue to prioritize the use of the numbered argument over that of the argM.

1.4.5 Manner (MNR)

Manner adverbs specify how an action is performed. For example, 'works well' is a manner. Manner tags should be used when an adverb could be an answer to a question starting with 'how?'.

*Among 33 men who-4 [*T*-4] worked closely with the substance, 28 [*ICH*-1] have died – more than three times the expected number.*
LINK-SLC: who-4 * 33 men
ARG0: [*T*-4]
REL: worked

ARGM-MNR: closely
ARG1: with the substance

Workers dumped large burlap sacks of the imported material into a huge bin, poured in cotton and acetate fibers and mechanically mixed the dry fibers in a process used [*] [*] to make filters.
ARG0: Workers
ARGM-MNR: mechanically
REL: mixed
ARG1: the dry fibers
ARGM-LOC: in a process used [*] [*] to make filters

1.4.6 Temporal (TMP)

Temporal ArgMs show when an action took place, such as 'in 1987', 'last Wednesday', 'soon' or 'immediately'. Also included in this category are adverbs of frequency (eg. often always, sometimes (with the exception of 'never', see NEG below), adverbs of duration (for a year/in an year), order (eg. first), and repetition (eg. again)..

[A form of asbestos]-2 once used [-2] [*] to make Kent cigarette filters has caused a high percentage of cancer deaths among a group of workers exposed [*] to it more than 30 years ago , researchers reported [0] [*T*-1] .*
ARG1: [*-2]
ARGM-TMP: once
REL: used
ARG2: [*] to make Kent cigarette filters

Four of the five surviving workers have asbestos-related diseases, including three with recently diagnosed cancer.
ARGM-TMP: recently
REL: diagnosed
ARG2: cancer

1.4.7 Extent (EXT)

ArgM-EXT indicate the amount of change occurring from an action, and are used mostly for the following:

```
-numerical adjuncts like '(raised prices) by 15%',
-quantifiers such as 'a lot'
-and comparatives such as '(he raised prices) more than she did'
```

*PS of New Hampshire shares closed yesterday at $ 3.75 [*U*], off 25 cents, in New York Stock Exchange composite trading.*
ARG1: PS of New Hampshire shares
REL: closed
ARGM-TMP: yesterday
ARGM-EXT: at $ 3.75 [*U*], off 25 cents,
ARGM-LOC: in New York Stock Exchange composite trading

*'An active 55-year-old in Boca Raton may care more about Senior Olympic games, while a 75-year-old in Panama City may care more about a seminar on health,' she says [*T*-1].*
ARG0: An active 55-year-old in Boca Raton
ARGM-MOD: may
REL: care
ARGM-EXT: more
ARG1: about Senior Olympic games
ARGM-ADV: while a 75-year-old in Panama City may care more about a seminar on health

Rep. Jerry Lewis , a conservative Californian , added a provision of his own, intended [*] to assist Bolivia, and the Senate then broadened the list further by [-1] including all countries in the U.S. Caribbean Basin initiate as well as the Philippines - [*-1] backed [*] by the powerful Hawaii Democrat Sen. Daniel Inouye.*
ARG0: the Senate
ARGM-TMP: then
REL: broadened
ARG1: the list
ARGM-EXT: further
ARGM-MNR: by [*-1] including all countries in the U.S. Caribbean Basin initiate as well as the Philippines
ARGM-PRD: [*-1] backed [*] by the powerful Hawaii Democrat Sen. Daniel Inouye

1.4.8 Reciprocals (REC)

These include reflexives and reciprocals such as himself, itself, themselves, each other, which refer back to one of the other arguments. Often, these arguments serve as the Arg 1 of the relation. In these cases, the argument should be annotated as the numbered argument as opposed to the reciprocal modifier.

*But voters decided that if the stadium was such a good idea someone would build it himself, and rejected it 59% to 41% [*U*].*
ARGM-ADV: if the stadium was such a good idea
ARG0: someone
ARGM-MOD: would
REL: build
ARG1: it
ARGM-REC: himself

1.4.9 Secondary Predication (PRD)

These are used to show that an adjunct of a predicate is in itself capable of carrying some predicate structure.

Typical examples include. . .

-Resultatives: as in 'The boys pinched them DEAD' or 'She kicked [the locker lid]-1 [*-1] SHUT'
-Depictives: 'ROSY-CHEEKED, Santa came down the chimney'
-as-phrases, e.g. 'supplied AS SECURITY IN THE TRANSACTION'

In each of these cases, it is notable that the argument labeled PRD modifies another argument of the verb (describing its state during or after the event) more than it modifies the verb or event itself.

Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .
ARG0: Pierre Vinken , 61 years old ,
ARGM-MOD: will
REL: join
ARG1: the board
ARGM-PRD: as a nonexecutive director
ARGM-TMP: Nov. 29

Prior to his term , a teacher bled to death in the halls , [-1] stabbed [*-2] by a student.*
ARGM-TMP: Prior to his term
ARG1: a teacher
REL: bled
ARGM-PRD: to death
ARGM-LOC: in the halls
ARGM-ADV: [*-1] stabbed [*-2] by a student

*This wage inflation is bleeding the NFL dry, the owners contend [*T*-1].*
ARG0: This wage inflation
REL: bleeding
ARG1: the NFL
ARGM-PRD: dry

[-2] Glamorous and pure-voiced as ever , Ms. Collins sang Joni Mitchell 's 'For Free' – about an encounter with a street-corner clarinetist , to which Mr. Stoltzman contributed a clarinet obligatto [*T*-1] – and Mr. Douglas 's lush setting of a Gaelic blessing , 'Deep Peace . '*
ARGM-PRD: [*-2] Glamorous and pure-voiced as ever
ARG0: Ms. Collins
REL: sang
ARG1: Joni Mitchell 's ' For Free ' – about an encounter with a street-corner clarinetist , to which Mr. Stoltzman contributed a clarinet obligatto [*T*-1] – and Mr. Douglas 's lush setting of a Gaelic blessing , ' Deep Peace.'

1.4.10 Purpose Clauses (PRP)

Purpose clauses are used to show the motivation for some action. Clauses beginning with 'in order to' and 'so that' are canonical purpose clauses.

More than a few CEOs say [0] the red-carpet treatment tempts them to return to a heartland city for future meetings.
ARG1: them
REL: return
ARG4: to a heartland city
ARGM-PRP: for future meetings

*In a disputed 1985 ruling , the Commerce Commission said [0] Commonwealth Edison could raise its electricity rates by $ 49 million [*U*] [*-1] to pay for the plant.*

ARG0: Commonwealth Edison
ARGM-MOD: could
REL: raise
ARG1: its electricity rates
ARG2: by $ 49 million [*U*]
ARGM-PRP: [*-1] to pay for the plant

1.4.11 Cause Clauses (CAU)

Similar to 'Purpose clauses', these indicate the reason for an action. Clauses beginning with 'because' or 'due to' are canonical cause clauses. Questions starting with 'why,' which are always characterized by a trace linking back to this question word, are always treated as cause. However, in these question phrases it can often be difficult or impossible to determine if the 'why' truly represents purpose or cause. Thus, as a general rule, if the annotator cannot determine whether an argument is more appropriately purpose or cause, cause is the default choice.

*Pro-forma balance sheets clearly show why-1 Cray Research favored the spinoff [*T*-1] .*
ARGM-CAU: [*T*-1]
ARG0: Cray Research
REL: favored
ARG1: the spinoff

*However , five other countries – China , Thailand , India , Brazil and Mexico – will remain on that so-called priority watch list because of an interim review , U.S. Trade Representative Carla Hills announced [0] [*T*-1] .*
ARGM-DIS: However
ARG1: five other countries – China , Thailand , India , Brazil and Mexico –
ARGM-MOD: will
REL: remain
ARG3: on that so-called priority watch list
ARGM-CAU: because of an interim review

1.4.12 Discourse (DIS)

These are markers which connect a sentence to a preceding sentence. Examples of discourse markers are: also, however, too, as well, but, and, as we've seen before, instead, on the other hand, for instance, etc. Additionally, vocatives wherein a name is spoken (e.g., 'Alan, will you go to the store?') and interjections (e.g., 'Gosh, I can't believe it') are treated as discourse modifiers. Because discourse markers add little or no semantic value to the phrase, a good rule of thumb for deciding if an element is a discourse marker is to think of the sentence without the potential discourse marker. If the meaning of the utterance is unchanged, it is likely that the element can be tagged as discourse.

Note that conjunctions such as 'but', 'or', 'and' are only marked in the beginning of the sentence. Additionally, items that relate the instance undergoing annotation to a previous sentence such as 'however,' 'on the other hand,' 'also,' and 'in addition,' should be tagged as DIS. However, these elements can alternately be tagged as ADV when they relate arguments within the clause being annotated (e.g. 'Mary reads novels in addition to writing poetry.') as opposed to relating to or juxtaposing an element within the sentence to an element outside the sentence (e.g. 'In addition, Mary reads novels'). Often, but not always, when these elements connect the annotation instance

to a previous sentence, they occur at the beginning of the instance. Consider these examples to clarify this difference:

But for now , they 're looking forward to their winter meeting – Boca in February.
ARGM-DIS: But
ARGM-TMP: for now
ARG0: they
REL: [looking] [forward]
ARG1: to their winter meeting – Boca in February

The notification also clarifies the requirements of the evaluation.
ARG0: The notification
ARGM-DIS: also
REL: clarifies ARG1: the requirements of the evaluation.

The notification recognizes the company and also clarifies the requirements of the evaluation.
ARG0: The notification
ARGM-ADV: also
REL: clarifies
ARG1: the requirements of the evaluation.

Remember, do not mark 'and', 'or', 'but', when they connect two clauses in the same sentence.

As previously mentioned, another type of discourse marker includes vocatives, which are marked as VOC in TreeBank:

TreeBank annotation:

```
(S (NP-VOC Kris),
    (NP-SBJ *)
    (VP go
        (ADVP-DIR home)))
```

PropBank annotation:
ARGM-DIS: Kris
REL: go
ARG0: [*]
ARGM-DIR: home

Vocative NPs in imperative sentences as shown above should not be tagged as coreference chains (i.e. Arg0: [*] * [Kris]) in order to make annotation consistent with other examples of Vocative NPs, which do not include traces:

I aint kidding you, Vince
ARGM-DIS: Vince
REL: kidding
ARG0: I
ARG1: you
ARGM-NEG: nt

And, finally, the class of Discourse markers includes interjections such as 'oh my god', 'ah', and 'damn.'

I might point out that your inability to report to my office this morning has not ah limited my knowledge of your activities as you may have hoped.
ARGM-DIS: ah
REL: limited
ARGM-NEG: not
ARG1: my knowledge of your activities
ARG0: your inability to report to my office this morning
ARGM-ADV: as you may have hoped

1.4.13 Modals (MOD)

Modals are: will, may, can, must, shall, might, should, could, would. These elements are consistently labeled in the TreeBank as 'MOD.' These are one of the few elements that are selected and tagged directly on the modal word itself, as opposed to selecting a higher node that contains the lexical item.

1.4.14 Negation (NEG)

This tag is used for elements such as 'not', 'n't', 'never', 'no longer' and other markers of negative sentences. Negation is an important notion for PropBank annotation; therefore, all markers which indicate negation should be marked as NEG. For example, when annotating adverbials like never, which could be marked as either TMP or NEG, the NEG tag should be used. These are also elements that are tagged directly on the lexical item itself as opposed to on a higher node. Be careful to distinguish these from the conjunction 'not only,' which does not actually indicate that the verb is negative and should not be annotated because it is a conjunction.

1.4.15 Direct Speech (DSP)

A verb of expression is any verb which has a speaker/thinker argument (Arg0) and the utterance/thought (Arg1). If the utterance is a constituent, then there is a trace in TreeBank which is coindexed with that constituent. PropBank annotation tags the trace as Arg1 in this case:

TreeBank Annotation:

```
(S ``
     (S-TPC-1 (NP-SBJ We)
              (VP will
                  (VP win)))
     ,
     ''
     (NP-SBJ Mary)
     (VP said
         (S *T*-1))
     .))
```

PropBank Annotation:
REL: said
ARG1: [*T*-1]
ARG0: Mary

Unfortunately, in many examples, the utterance does not correspond to one constituent in TreeBank:

Among other things , they said [?*] , Mr. Azoff would develop musical acts for a new record label.*

TreeBank Annotation:

```
(S
    (PP (IN Among)
      (NP (JJ other) (NNS things) ))
    (PRN
      (, ,)
      (S
        (NP-SBJ (PRP they) )
        (VP (VBD said)
          (SBAR (-NONE- 0)
            (S (-NONE- *?*) ))))
      (, ,) )
    (NP-SBJ (NNP Mr.) (NNP Azoff) )
    (VP (MD would)
      (VP (VB develop)
        (NP
          (NP (JJ musical) (NNS acts) )
          (PP (IN for)
            (NP (DT a) (JJ new) (NN record) (NN label) )))))
    (. .) )
```

As the example above shows, in such cases, the Arg1 argument of the verb say is a *?* empty category, which does not have an index in TreeBank. PropBank annotation tags this empty category as Arg1 in this case; however, it also provides a link between this empty category and the top S node, which contains the utterance as well as the verb of saying. This is a rare exception wherein the annotation will include embedded arguments. Thus, the annotator first selects and tags the entire 'SBAR' node as Arg 1. Next, the annotator selects the empty trace itself 'NONE-*?*' and annotates this node as ARGM-DSP. While this annotation is still in the Jubilee memory, the annotator subsequently selects the S-node containing the rel and uses the '*' link to link the two together: click `Argument` on the Jubilee menu bar followed by clicking `Functions`. From the options therein, select '*' (shortcut: Ctrl+Shift-8). Although seasoned annotators should feel uncomfortable creating embedded, recursive annotations such as this, there is special post-processing for these cases that effectively removes what is often a PRN (parenthetical) node containing the relation and its arguments from the argument that is semantically 'what is spoken.' In the final version of the PropBank, ArgM-DSP tag will be replaced by LINK-DSP, to indicate that this is not a modifier of the verb, but simply additional information about one of its arguments.

ARG1: [SBAR (-NONE- 0) S (-NONE- *?*)]
ARG0: they
ARGM-DSP: [-NONE-*?*] * [Among other things , they said [*?*] , Mr. Azoff would develop musical acts for a new record label]
rel: said

Figure 1.1 shows the correct annotation of an instance of DSP as it will be seen in Jubilee.

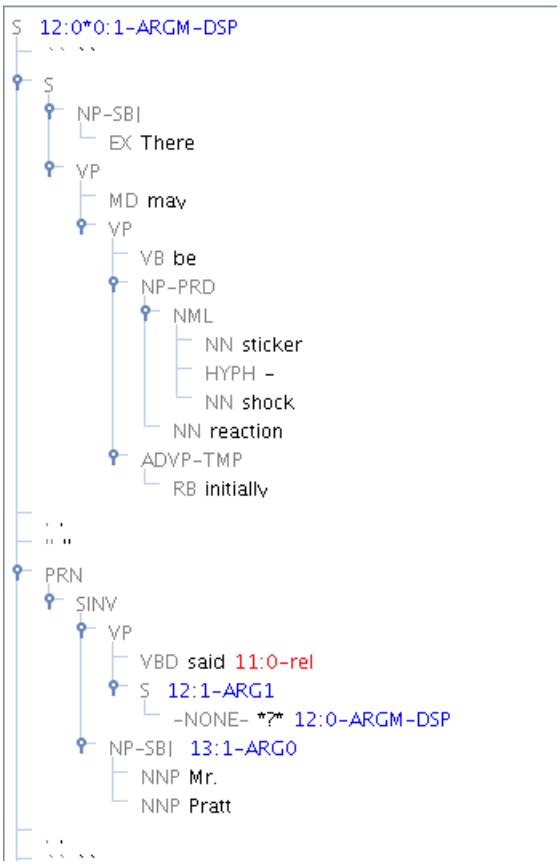

Figure 1.1: Correct annotation of DSP

1.4.16 Light Verb (LVB)

This tag is used to label the light verb only in the noun pass of light verb annotation. See Chapter 2 for more details.

1.4.17 Adverbials (ADV)

These are used for syntactic elements which clearly modify the event structure of the verb in question, but which do not fall under any of the headings above. The annotator should always try to use one of the alternate modifiers listed above before assuming that a modifier is merely adverbial. However, adverbial elements are often. . .

```
1. Temporally related (modifiers of events)
Treasures are just lying around, waiting to be picked up
2. Intensional (modifiers of propositions)
Probably, possibly
3. Focus-sensitive
Only, even
4. Sentential (evaluative, attitudinal, viewpoint, performatives)
Fortunately, really, legally, frankly speaking,
clauses beginning with ‘given that’, ‘despite’, ‘except for’, ‘if’
```

As opposed to ARGM-MNR, which modify the verb, ARGM-ADVs usually modify the entire sentence.

In some cases, modifiers like happily can be ambiguous between MNR and ADV interpretations, as shown below:

She sang happily.
ARGM-MNR: happily

Happily, she sang. (paraphrasable as 'I am happy that she sang')
ARGM-ADV: happily

In these cases, use context as much as possible to try to make the best judgment.

1.4.18 Adjectival (ADJ)

This tag is used in a manner that is similar to ADV, but its use is restricted to noun annotation. See Chapter 3 for more details.

1.5 Span of Annotation

For the purposes of PropBank annotation, annotators should only assign arguments within a certain syntactic span surrounding the REL. The structure of the tree reflects which constituents in an utterance are truly arguments of a particular predicate; thus, even when annotators feel that a constituent outside of this span has some semantic bearing on the REL, it should not be annotated. Rather, the syntactic span of annotation should be respected: everything within that span should be encompassed by an argument label (with exceptions described below), and nothing outside of that span should be annotated (with exception of linking annotation, such as that of relative clauses).

Do not tag determiners (labeled DT in TreeBank) or conjuctions (labeled CC or CONJP in TreeBank), unless these begin the sentence and are being used in a discourse function, as described in section 1.4.12. Do not tag auxiliary verbs such as 'have,' 'be' or 'do'; the auxiliary verb itself will come up for annotation and at that point the auxiliary sense will be selected without further annotation.

```
Tag all and only the following:
 Sisters of the verb
 Sisters of the VP
```

To determine the span of annotation, locate the rel and the accompanying TreeBank tag indicating one of the following types of verbs [1]:

VB Verb, base form
VBD Verb, past tense
VBG Verb, gerund or present participle
VBN Verb, past participle
VBP Verb, non-3rd person singular present
VBZ Verb, 3rd person singular present

Figure 1.2 shows the TreeBank view of a typical instance in Jubilee, with the VB node indicated.

Once this has been located, annotate the sisters of this node. Sisters to the verb will be parallel to it in the tree. Figure 1.3 has an arrow where the annotator should look, beginning at the

[1] For a complete listing of TreeBank tags, see http://bulba.sdsu.edu/jeanette/thesis/PennTags.html

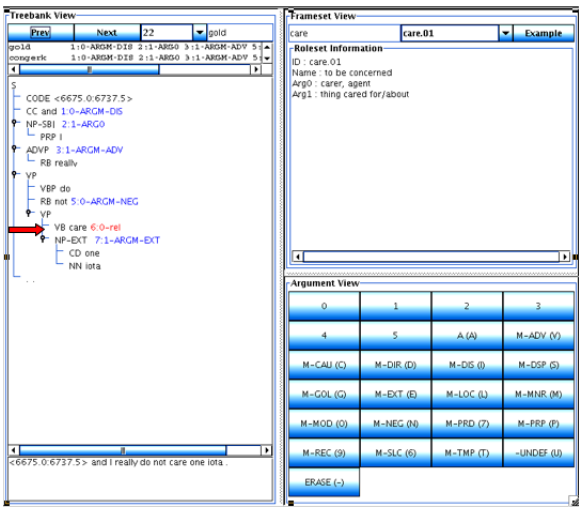

Figure 1.2: Find the rel

verb, for sisters. In this case, it is a relatively short distance and only the NP-EXT node needs to be annotated as ARG-1; however, in some cases, the annotator will have to scroll up and down through Jubilee's TreeBank view to annotate multiple sisters to the verb.

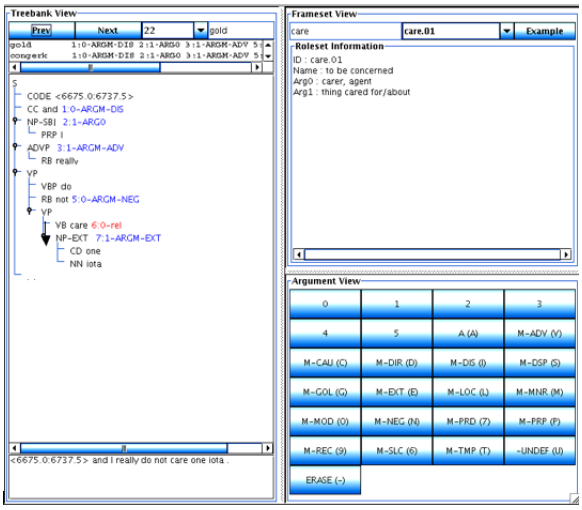

Figure 1.3: Find and annotate any sisters to the rel

Next, examine the tree to see if the VB node is embedded in a VP (verb phrase) node. The verb is usually located inside a higher VP node, unless it is located inside a NP (noun phrase) node. Where the verb is accompanied by one or more auxiliaries, it may be encompassed by several VP nodes, as illustrated by Figure 1.4, which indicates each of the VP nodes:

Once the annotator has located the highest VP node, annotate all sisters to each VP node (again, the nodes that are parallel to VP nodes in the tree). Figure 1.5 uses arrows to illustrate where the annotator should look, beginning at the verb, for sisters to the VP nodes:

Thus, following this line, the annotator discovers several nodes that must be annotated: firstly, the RB node as ARGM-NEG; the auxiliary 'do' can be ignored; second, the ADVP node as ARGM-ADV; next the NP-SBJ node as ARG-0; finally the CC node as ARGM-DIS because it meets the conditions described in section 1.4.12 for serving a discourse function. The CODE node can be ignored when present. Additionally 'TOP' nodes at the very top of an instance should never be annotated.

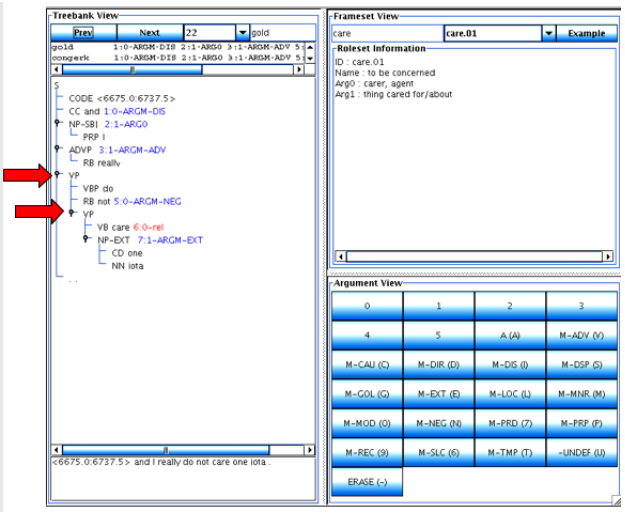

Figure 1.4: Find the highest VP node containing the rel

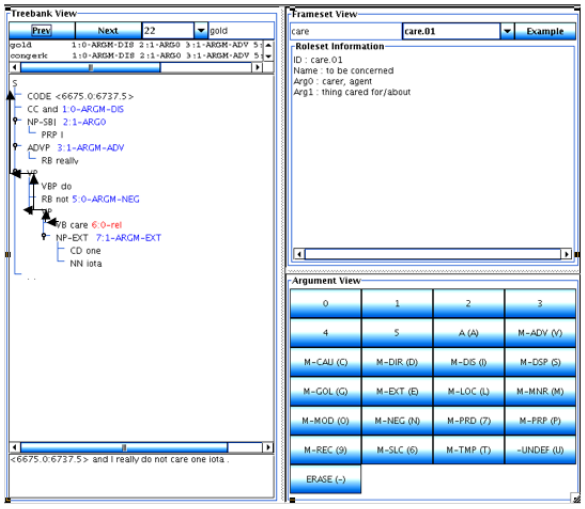

Figure 1.5: Find and annotate any sisters to the VP nodes containing the rel

The last thing to note is that when the verb is embedded in a VP node, 'S marks the spot' to stop annotation:

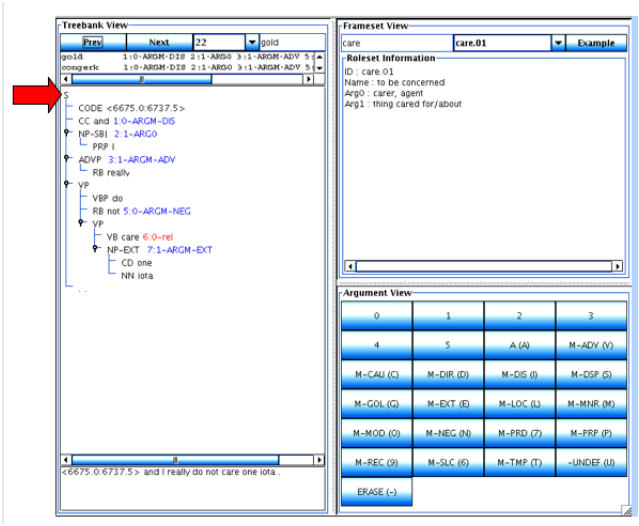

Figure 1.6: Stop at S

'S' indicates clausal boundaries in the TreeBank. Thus, anything beyond the S would also be beyond the clause containing the REL, and in turn, constituents outside of this clause are not arguments of the REL. Only linking practices should require attention to constituents outside of the S node containing the REL. TreeBank annotation can mark clausal boundaries with any of the following tags, indicating what type of clausal boundary it is; all should be treated as marking one end of the annotation span:

S - simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a wh-word and that does not exhibit subject-verb inversion.
SBAR - Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ - Direct question introduced by a wh-word or a wh-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.
SINV - Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ - Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ.

Note that S nodes can also serve as sentential complements to a verb, as seen in Figure 1.7. Which S node is of focus when determining the span depends on the relative location of the rel.

As mentioned previously, the REL can potentially arise in other types of nodes, such as NP (noun phrase) or ADJP (adjective phrase) nodes, with or without an intervening VP node. When this happens, the annotation span cannot be thought of as delimited by an S node. Just as in previously mentioned cases, annotate sisters to the verb or VB TreeBank tag and annotate sisters to the VP node when present. Although the S node will not be present as a clue for where to stop annotation, simply do not annotate constituents that are parents or aunts to the verb or verb phrase node. Parents and aunts, unlike sisters, will not be along a parallel line with the verb or verb phrase. Instead, their root nodes will be located to the left of this line. This heuristic applies when determining the span of any annotation, but may be especially important in the absence of an S node.

Figure 1.8 gives an example of a rel contained within an NP node. Note that the NP node

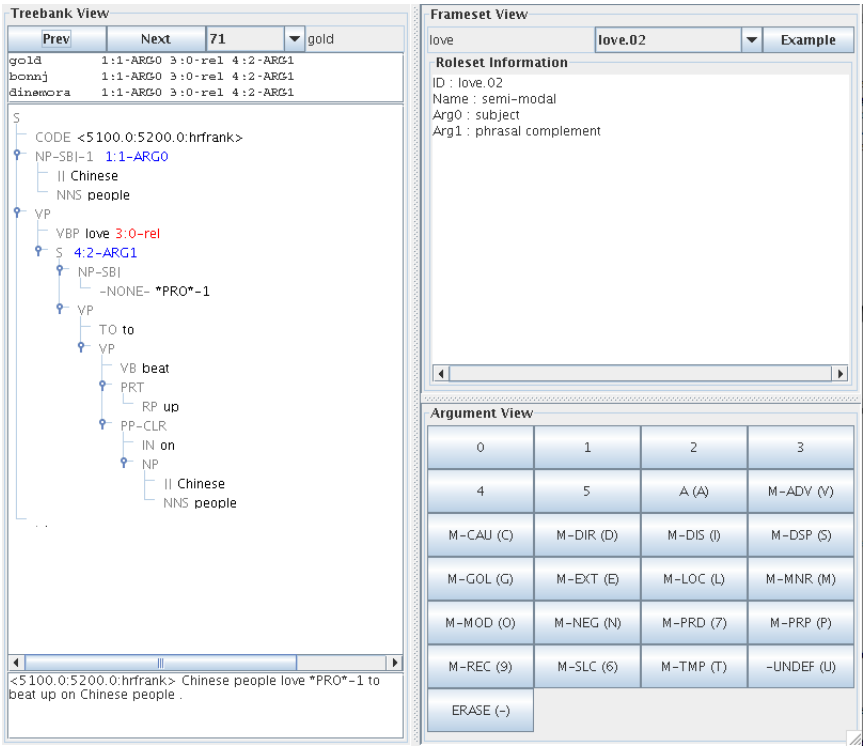

Figure 1.7: S as sentential complement

itself and everything outside of this node should not be annotated because these constituents are parents or aunts of the verb phrase rather than sisters of the verb phrase or verb.

1.6 Where to place tags

Jubilee allows you to select any node in the tree; thus, it is up to the annotator to select the appropriate node reflecting the correct constituent boundaries of an argument. In general, the node above the lexical item itself, which indicates the syntactic function of that constituent (e.g., NP or NP-SBJ, PP, ADJP, ADVP, etc.), is the correct placement for the tag. However, as mentioned in the Arg-M sections, annotation of modals and negatives require placement of the tag directly on the lexical item because there is no higher node to annotate without including more than just the modal or negative marker. Please review the Jubilee screen shots given in section 1.5 for examples of the correct placement of tags.

Occasionally, the phrase structure of an instance is such that the annotator must choose between annotating a higher node as a single argument or annotating several nodes embedded therein as various arguments. As a general rule of thumb, if it is possible to place a lower-numbered argument tag on a single, higher node, this is preferable to annotating several higher-numbered arguments and/or modifiers on embedded nodes therein. For example, figure 1.9 below is correct given the argument structure outlined in the frame hold. 04, but it is dispreferred to figure 1.10, which simply tags the higher node with a lower argument number.

1.6.1 Exceptions to normal tag placement

Certain verbs such as ‘encourage’ and ‘persuade,’ which involve both an impelled agent and an impelled action require that annotators ‘break up’ and delve into the sister S-node in order to annotate the impelled agent and impelled action separately. These are verbs that participate in

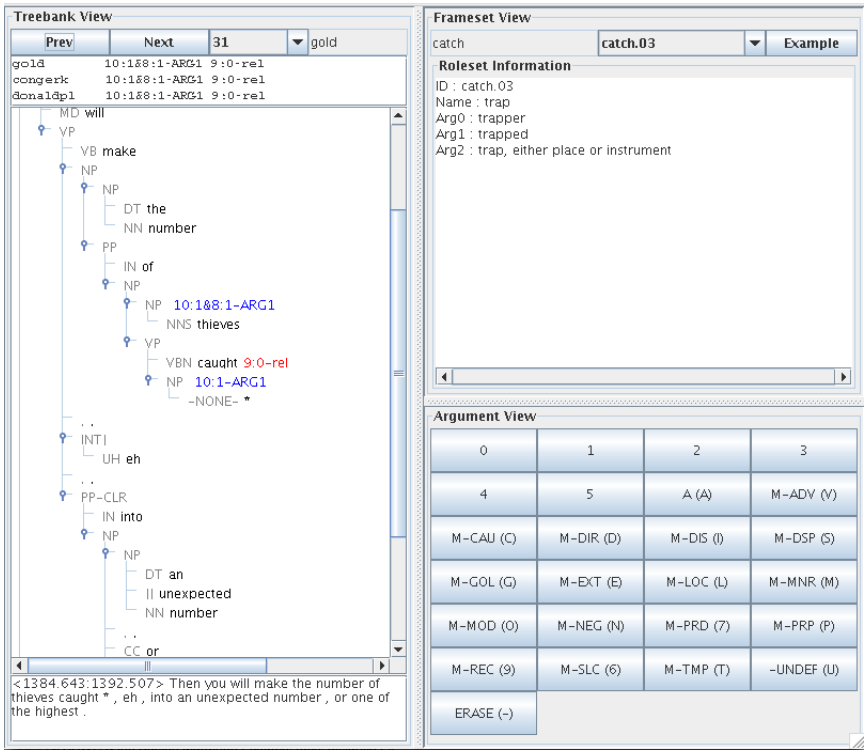

Figure 1.8: Rel embedded in an NP node

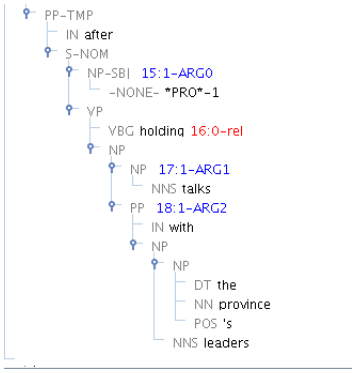

Figure 1.9: Theoretically correct, but dispreferred annotation

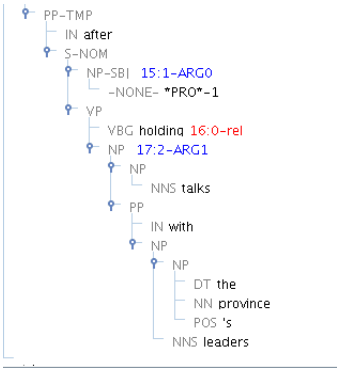

Figure 1.10: Preferred annotation of higher node with lower argument number

exceptional case marking, meaning that the matrix verb (e.g. 'encourage' or 'persuade') assigns the accusative case to the subject of the sentential complement:

I encouraged HIM to annotate for PropBank.
*I encouraged he to annotate for Propbank.

Because the matrix verb assigns accusative case to the subject of the infinitival complement, it is thought that the verb assigns semantic roles to both the agent and action separately. However, the sentential complement of these verbs forms a single constituent, which is an S-node that is a sister to the matrix verb. As a result, where we would normally only annotate the S-node with an argument, in these cases, we annotate within the S node to separately tag the impelled agent and impelled action.

1.7 Understanding and annotating null elements in the Penn TreeBank

The inventory of null elements used in Penn TreeBank is as follows:

```
[*PRO*] (overt subjects, subject control, and small clauses)
[*] (passive traces including reduced relative clauses and raising constructions)
[*T*]      (trace of A-movement, including parasitic gaps)
[(NP *)]   (arbitrary PRO, controlled PRO, and trace of A-movement)
[0]        (null complementizer, including null wh-operator)
[*U*]      (unit)
[*?*]      (placeholder for ellipsed material)
[*NOT*]    (anti-placeholder in template gapping)
[*RNR*]    (pseudo-attach: right node raising)
[*ICH*]    (pseudo-attach: interpret constituent here)
[*EXP*]    (pseudo-attach: extraposition)
```

This section presents some examples of most commonly used null elements and their PropBank annotation.

1.7.1 Passive Sentences

Sentences can be either active (The executive committee approved the new policy) or passive (The new policy was approved by the executive committee). In active sentences, the subject is the agent or a do-er of the action, marked as Arg0 in PropBank. In passive sentences, the subject of the sentence is acted upon by some other agent or by something unnamed, and is being marked as Arg1 in PropBank.

Passive sentences are assumed to be derived from the corresponding active sentences by movement of the object to the subject position. This movement leaves a trace, represented as [*] in TreeBank. Except in the case of reduced relatives, this trace will already be coindexed with its realized referent:

Active: Mary hit John
Passive: John-1 was hit [*-1] by Mary.

Since TreeBank provides a link between [*-1] and John, it is the TRACE, rather than the NP 'John,' which is being labeled as Arg1 in PropBank:

PropBank annotation:
REL: hit

ARG1: [*-1]
ARG0: by Mary

The following example illustrates a TreeBank representation of the passive sentence. The link between the trace and the NP is indicated by the number 1 in the trace (NP-3 *-1) and (NP-SBJ-1 he) below. Note that chains of coreference are represented in the Penn TreeBank using various numerals, and that one element such as 'he' can potentially be semantically present in several positions in the underlying syntax; therefore, several numbered indices may be connected to one element. It is important to follow the chains of coreference throughout the instance to ensure a full understanding of each null element.

TreeBank annotation:

```
(S (NP-SBJ-1 he)
   (VP was
       (VP accused
           (NP-3 *-1)
           (PP-CLR of
                   (S-NOM (NP-SBJ *-3)
                          (VP (VP conducting
                                  (NP illegal business))
                              and
                              (VP possessing
                                  (NP illegal materials))))))))
```

Again, it is the trace which is being annotated as the argument:

PropBank annotation:
ARG1: [NP-3 *-1]
REL: accused
ARG2: of [*3*] conducting illegal business and possessing illegal materials

1.7.2 Fronted and Dislocated Arguments

Other examples of moved constituents are fronted or otherwise dislocated arguments and adjuncts. As in the other cases of movement, fronted elements leave a trace, which is being coindexed with the moved constituent in TreeBank.

In the following example, the Arg2 ('where put') argument of the verb 'put' is being fronted. In the TreeBank annotation, this is indicated by the chain which links the trace [*T*-1] with the adverbial 'There':

TreeBank annotation:

```
(S (ADVP-PUT-TPC-1 There)
   ,
   (NP-SBJ I)
   (VP put
       (NP the book)
       (ADVP-PUT *T*-1) ))
```

As with annotation of passive traces, the Arg2 argument is the TRACE, rather than the fronted constituent:

PropBank annotation: REL: put
ARG0: I
ARG1: the book
ARG2: [*T*-1]

Modifiers, or ArgMs, can be fronted as well, as the following example shows:

TreeBank annotation:

```
(S (SBAR-PRP-TPC-9 Because
                      (S (NP-SBJ I)
                         (VP 'm
                             (NP-PRD such a bad boy))))
     (NP-SBJ I)
     (VP think
         (SBAR 0
               (S (NP-SBJ I)
                  (VP wo n't
                      (VP get
                          (NP a lollipop)
                          (SBAR-PRP *T*-9) )))))
```

Since the 'because' clause modifies the verb 'get' in this example, the trace originates as the modifier of 'get'. This trace is being annotated as ArgM-CAU in PropBank:

PropBank annotation:
REL: get
ARG1: a lollipop
ARG0: I
ARGM-NEG: n't
ARGM-MOD: wo
ARGM-CAU: [*T*-9]

In rare situations, movement does not leave a trace, but rather leaves a pronoun (called a resumptive pronoun). In such cases, the argument of the verb is a higher NP, which includes both the pronoun and the trace to the topicalized NP in TreeBank. This NP is annotated as Arg1 in PropBank:

TreeBank annotation:

```
  (S (NP-TPC-1 John)
     ,
     (NP-SBJ I)
     (VP like
         (NP (NP him)
             (NP-1 *T*))
         (NP-ADV a lot)))
```

PropBank annotation:
REL: like
ARG0: I
ARG1: [NP (NP him) (NP-1 *T*)]
ARGM-MNR: a lot

In even more rare situations, the topicalized NP and the pronoun are not already co-indexed in the TreeBank. See section 1.8.4 for further description of how to annotate these instances.

1.7.3 Questions and Wh-Phrases

Another type of traces is a trace of a wh-phrase in questions.

What do you like?

As in the case of passive sentences, questions are assumed to be derived by movement. In the example below, the Arg1 argument of the verb 'like' is a wh-phrase 'what', which moves from the object position of the verb to the front of the sentence. This movement leaves a trace, as shown below:

*What-1 do you like [*T*-1]?*

In TreeBank annotations, wh-phrases are marked as WHNP. As in the case of passive sentences, TreeBank provides a link between the trace and the moved WHNP:

TreeBank annotation:

```
  (SBARQ (WHNP-1 what)
         (SQ do
             (NP-SBJ you)
             (VP like
                  (NP *T*-1)))
```

Again, for the purposes of PropBank, the argument Arg1 is the trace, as shown below:

PropBank annotation:
REL: like
ARG0: you
ARG1: [*T*-1]

Wh-phrases are not necessarily core arguments. However, questions can be formed with wh-phrases like when, where, how, in which case they should be tagged as ArgMs.

TreeBank annotation:

```
SBARQ (WHNP-1 Which day)
          (SQ did
              (NP-SBJ you)
              (VP get
                  (ADVP-DIR there)
                  (NP-TMP *T*-1)))
```

PropBank annotation:
ARG0: you
REL: get
ARG2: there
ARGM-TMP: [*T*-1]

TreeBank annotation:

```
  (SBARQ (WHADVP-42 How)
         (SQ did
```

```
            (NP-SBJ you)
            (VP fix
                (NP the car)
                (ADVP-MNR *T*-42)))
        ?)
```

PropBank annotation:
REL: fix
ARG0: you
ARG1: the car
ARGM-MNR: [*T*-42]

Questions can also be embedded, as in the example below. PropBank annotation is not different from direct questions in this case:

*John didn't know where-3 his parents had met [*T*-3].*

ARG0: his parents
REL: met
ARGM-LOC: [*T*-3]

1.7.4 ICH Traces (ICH: interpret constituent here)

ICH traces are used in TreeBank to indicate a relationship of constituency between elements separated by intervening material. An example of such split constituents are 'heavy shift' constructions, illustrated below:

TreeBank annotation:

```
(S (NP-SBJ (NP a young woman)
           (SBAR *ICH*-1))
   (VP entered
       (SBAR-1 (WHNP-2 whom)
               (S (NP-SBJ she)
                  (PP-TMP at
                          (ADVP once))
                  (VP recognized
                      (NP *T*-2)
                      (PP-CLR as
                              (NP Jemima Broadwood)))))))
```

The subject NP in this case is being split into two constituents: the NP 'a young woman' and SBAR: 'whom she at once recognized as Jemima Broadwood'. The ICH trace specifies a link to the SBAR node in this example. Essentially, the NP in addition to the material linked by ICH trace can be thought of as one whole constituent: 'A young woman whom she at once recognized as Jemima Broadwood,' part of which has been moved for pragmatic purposes.

In all examples of this type, the argument is the constituent which includes the ICH trace:

PropBank annotation:
ARG0: a young woman [*ICH*-1]
REL: entered

It is very important that the annotator does not annotate the dislocated part of the constituent (in the previous case the SBAR-1 material) a second time with another tag. Underlyingly, the material connected by ICH trace is part of the NP 'a young woman,' which is already annotated as ARG-0. In post-processing, the rest of this constituent linked by ICH trace will be concatenated to the NP annotated as ARG-0; thus, tagging the dislocated portion of the constituent a second time will create recursive annotation and will be returned as an error. In other words, tagging the dislocated portion will be recognized computationally as having a second argument of a different type embedded in the first, which is disallowed.

Other typical examples of *ICH* traces are shown below:

*[Five *ICH*-1] ran, [out of the twenty-five that showed up]-1.*
ARG0: Five *ICH-1*
REL: ran

*[Some people in Paris]-1 want *PRO*-1 to hear more [*ICH*-2] from me [than those fellers over at the conference house do]-2.*
ARG0: *PRO*-1
REL: hear
ARG1: more [ICH-2]
ARG2: from me

Figure 1.11 shows correct annotation for an instance containing an ICH trace.

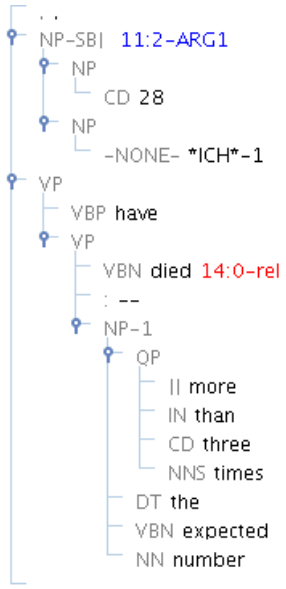

Figure 1.11: Correct annotation of ICH

1.7.5 Right Node Raising (RNR) Traces

RNR traces are used when a constituent is interpreted simultaneously in more than one place. An example of a right node raising structure is given below:

TreeBank annotation:

```
(NP (NP (ADJP so many) enchained demons)
      (VP straining
```

```
(PP-MNR in
        (NP anger))
(S (NP-SBJ *)
   (VP to
       (VP (VP tear
               (NP *RNR*-1))
           and
           (VP gnaw
               (PP-CLR on
                       (NP *RNR*-1)))
           (NP-1 his bones))))))
```

In this example, the NP 'his bones' is interpreted as both the argument of the verb 'tear' and the verb 'gnaw'. When annotating the verb 'tear', the trace (NP *RNR*-1) is the argument of the verb:

PropBank annotation:
REL: tear
ARG1: [*RNR*-1]
ARG0: [NP-SBJ*]

Likewise, when annotating the verb 'gnaw', the prepositional phrase including the trace (PP-CLR on (NP *RNR*-1))) is analyzed as the argument:

PropBank annotation:
REL: gnaw
ARG1: on [*RNR*-1]
ARG0: [NP-SBJ*]

A similar annotation applies when the [*RNR*] trace is a clausal argument:

*I want *RNR*-1 and like *RNR*-1 [* to eat ice-cream]-1.*
ARG0: I
REL: want
ARG1: *RNR*-1

If the RNR trace is part of the argument of the verb, then select the argument including the trace:

His dreams had revolved around her so much and for so long that...

TreeBank annotation:

```
(S (NP-SBJ His dreams)
   (VP had
       (VP revolved
           (PP-CLR around
                   (NP her))
           (UCP-ADV (ADVP (ADVP so much)
                          (SBAR *RNR*-1))
                    and
                    (PP-TMP for
```

```
                        (NP (NP so long)
                            (SBAR *RNR*-1)))
                (SBAR-1 that...)))))
```

PropBank annotation:
ARG1: his dreams
REL: revolved
ARGM-LOC: around her
ARGM-EXT: so much [*RNR*]

The following example illustrates annotation of RNR traces within a small clause (for further information on the annotation of small clauses, see section 1.9).

But our outlook has been and continues to be defensive

TreeBank annotation:

```
(S But
   (NP-SBJ-2 our outlook)
   (VP (VP has
           (VP been
               (ADJP-PRD *RNR*-1)))
       ,
       and
       (VP continues
           (S (NP-SBJ *-2)
              (VP to
                  (VP be
                      (ADJP-PRD *RNR*-1)))))
       ,
       (ADJP-PRD-1 defensive)))
```

PropBank annotation:
REL: continue
ARG1: [*-2] to be *RNR-1

Figure 1.12 shows correct annotation of an instance containing RNR traces.

1.7.6 *EXP* (It EXtraPosition)

Dummy placeholders in English such as 'it' or 'that' do not add any meaning to the sentence. In the following example, the syntactic subject of the sentence is a dummy 'it', which includes a trace EXP-1. This trace refers to the logical, semantic subject of the sentence, marked as SBAR-1:

TreeBank annotation:

```
(S (NP-SBJ (NP It)
           (SBAR *EXP*-1))
   (VP is
       (ADJP-PRD clear)
       (PP to
           (NP me))
       (SBAR-1 that
```

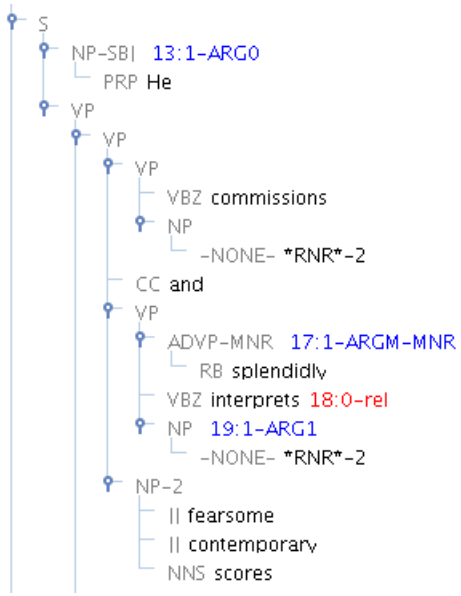

Figure 1.12: Correct annotation of RNR

```
(S (NP-SBJ this message)
   (VP is
       (ADJP-PRD unclear))))))
```

In PropBank annotations, dummy 'it' and EXP traces are NOT INCLUDED, DO NOT TAG THEM:

PropBank annotation:
REL: is
ARG1: that this message is unclear
ARG2: clear to me

Rather, tag only that which has semantic value in the utterance, the overt constituent. Thus, the underlying phrase can be thought of semantically as: 'That this message is unclear is clear to me.' The 'it' is merely added for pragmatic purposes to avoid having such a heavy constituent at the front of the phrase.

Another example:

It required an energy he no longer possessed to be satirical about his father.

PropBank annotation:
ARG0: to be satirical about his father
ARG1: an energy he no longer possessed
REL: required

In the examples below, the dummy constituents are the objects, rather than the subjects. As in the case of dummy subjects, only the logical argument is being tagged, whereas the dummy pronoun and the EXP trace are not part of the PropBank annotation:

Mrs. Yeargin was fired [-1] and prosecuted [*-1] under an unusual South Carolina law that-79 [*T*-79] makes it [*EXP*-2] a crime [*] to breach test security.*

PropBank annotation:

LINK-SLC: that-79 * an unusual South Carolina law
ARG0: [*T*-79]
REL: makes
ARG2: a crime
ARG1: [*] to breach test security

*Any raider would find it [*EXP*-1] hard [*] to crack AG 's battlements.*

TreeBank annotation:

```
(S
    (NP-SBJ (DT Any) (NN raider) )
    (VP (MD would)
      (VP (VB find)
        (S
          (NP-SBJ
            (NP (PRP it) )
            (S (-NONE- *EXP*-1) ))
          (ADJP-PRD (JJ hard) )
          (S-1
            (NP-SBJ (-NONE- *) )
            (VP (TO to)
              (VP (VB crack)
                (NP
                  (NP (NNP AG) (POS 's) )
                  (NNS battlements) )))))))
    (. .) )
```

PropBank annotation:
ARG0: Any raider
ARGM-MOD: would
REL: find
ARG3: hard
Arg1: [*] to crack AG 's battlements

Figure 1.13 shows correct annotation of an instance of the copular sense of 'to be,' which contains it *EXP*.

Common mistake: Please make sure to distinguish dummy 'it' from the referring pronoun 'it', where 'it' refers to a previous NP, a clause, or an event. (hint: referring pronouns are not followed by an EXP trace in TreeBank). All referring pronouns, including 'it', should be marked as arguments in PropBank.

It sounds good.
REL: sounds
ARG1: it
ARGM-MNR: good

Italy 's Foreign Ministry said [0] it is investigating exports to the Soviet Union.
REL: investigating
ARG0: it

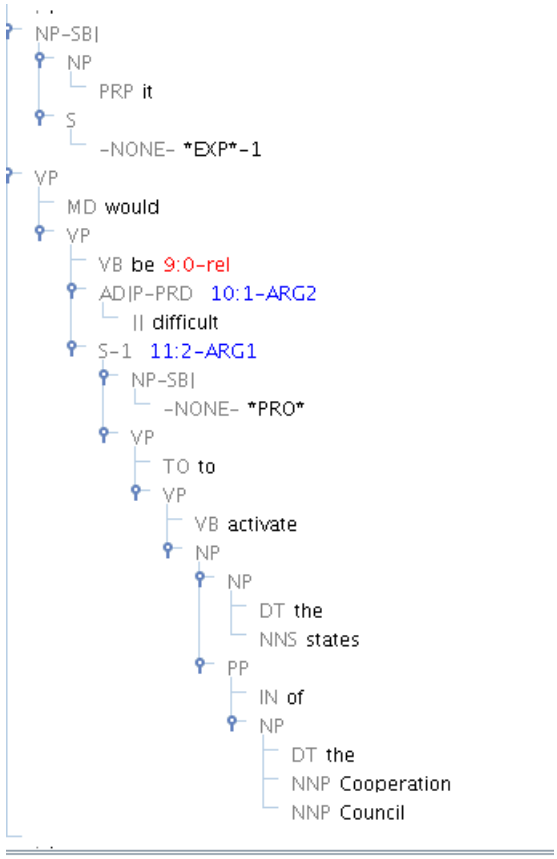

Figure 1.13: Correct annotation of EXP and be.01

ARG1: exports to the Soviet Union

1.7.7 Other Traces

Other types of traces include the null complementizer trace '0', the '?' trace (used in ellipsis constructions), and *PPA* trace in cases of predictable ambiguous attachments.

Null complementizer traces should be included as part of the clausal argument; thus, the ARG-1 in this case would be annotated at the level of the SBAR node:

TreeBank annotation:

```
(S (NP-SBJ I)
    (VP believe
        (SBAR 0
              (S (NP-SBJ you)
                 (VP are
                     (ADJP-PRD smart))))))
```

PropBank annotation:
REL: believe
ARG0: I
ARG1: [[0] you are smart]

1.8 Linking & Annotation of Null Elements

1.8.1 SeLectional Constraint Link (Link SLC)

Relative Clause Annotation

Typical Relative Clauses Relative clauses are clauses that modify a N or a NP as in 'answers that we'd like to have.' Relative clauses also include a trace, which is coindexed with the relativizer in TreeBank (e.g. 'that'/'which'/'who'). Alternatively, the relativizer can be omitted in English: 'answers we'd like to have.' In these cases the TreeBank will still include a placeholder for the relativizer, but a '0' will appear where the explicit relativizer normally appears.

For example, in the following TreeBank annotation, the object position of the verb has a trace (NP *T*-6), which is being coindexed with the relativizer (WHNP-6 that/which/0).

TreeBank annotation:

```
(NP (NP answers)
     (SBAR (WHNP-6 that/which/0)}
          (S (NP-SBJ-3 we)
             (VP 'd
                 (VP like
                     (S (NP-SBJ *-3)
                        (VP to
                            (VP have
                                (NP *T*-6)))))))))
```

Whereas, syntactically, the trace is being coindexed with the relativizer, semantically, there is a relationship between the trace and the NP 'answers,' which is not being represented in TreeBank. To capture this relationship, which is useful for many applications, PropBank annotators should add a link from the relativizer to the NP it is associated with (this NP is almost always syntactically a sister to the SBAR). Firstly, the relativizer should be annotated as LINK-SLC (selectional constraint link). Secondly, the annotator should select the appropriate NP node and create the link by clicking `Argument` on the Jubilee menu bar followed by clicking `Functions`. From the options therein, select '*' (shortcut: Ctrl+Shift-8). At this point, the linked annotation should appear on the selected NP node.

PropBank annotation:
ARG1: [NP *T*-6]
LINK-SLC: [WHNP-6 that/which/0] * [NP answers]
REL: have
ARG0: [NP-SBJ *-3]

Similarly, if a relative clause modifies a temporal or a locative, the trace of the temporal or locative is marked as ArgM-TMP or ArgM-LOC, and the appropriate relativizer 'when' or 'where' is annotated as ArgM-LINK-SLC and linked to an NP:

*John found the place where-3 his parents had met *T*-3.*
ARG0: his parents
REL: met
ARGM-LOC: [*T*-3]
LINK-SLC: where * the place

Relative clauses should be annotated in all cases where a trace that is coindexed with a relativizer (or empty relativizer '0') is a numbered or adjunct argument of the relation. This includes the possessive relativizer 'whose':

TreeBank annotation:

```
(NP (NP (NN men)
            (SBAR
              (WHNP-2
                  (WP whose)
                  (NNS noses)
                (S
                  (NP-SBJ
                    (PRP he)
                  (VP
                    (VBD rubbed)
                    (NP
                      (-NONE- *T*-2)
                    (PP-CLR
                      (IN in)
                      (NP
                        (DT the)
                        (NN ground))))))))
```

PropBank annotation:
REL: rubbed
ARG1: [-NONE- *T*-2]
LINK-SLC: [WHNP-2 whose noses] * [NP men]
ARG0: [NP-SBJ he]
ARG2: [PP-CLR in the ground]

Infinitival Relatives Although they are not true relative clause constructions, infinitival relatives are also treated in the same manner. Note that infinitival relatives will always have a '0' in the position of the relativizer:

TreeBank annotation:

```
(NP (NP a movie)
      (SBAR (WHNP-1 0)
            (S (NP-SBJ *)
               (VP to
                   (VP see
                       (NP *T*-1))))))
```

PropBank annotation:
REL: see
ARG1: [NP *T*-1]
ARG0: [NP-SBJ *]
LINK-SLC: [WHNP-1 0] * [NP a movie]

Exceptions: when to NOT annotate relative clauses However, if the trace node associated with a relativizer is not a sister to the relation, then the relative clause should NOT

be annotated. These will be cases, for example, where the trace coindexed with a relativizer is further embedded in a prepositional node, so it is a daughter to the relation rather than a sister:

TreeBank annotation:

```
(NP (NP (NN game shows)
        (SBAR
          (WHPP-1
            (IN of
            (WHNP
              (WDT which)
          (S
            (NP-SBJ
              (PRP It)
            (ADVP-TMP
              (RB recently)
            (VP
              (VBD bought)
              (NP (NP 49\%
                      (PP -NONE- *T*-1)))))))
```

PropBank annotation:
ARG0: [NP-SBJ It]
ARGM-TMP: [ADVP-TMP recently]
REL: bought
ARG1: [NP 49% -NONE- *T*-1]

In the above example, the relative clause is not annotated because the trace is not a sister to the verb, rather it is embedded in a PP node within an NP node that is an argument of the verb.

Additionally, the relative clause should NOT be annotated when the relation is the subordinate verb in an infinitival complement because, in these cases, the relative clause does not syntactically modify the verb of the complement clause:

*The man that-1 *T*-1 wanted *PRO*-1 to leave home.*
RELl: wanted
ARG0: *T*-1
LINK-SLC: that * The man
ARG1: *PRO*-1 to leave home

As opposed to. . .

*The man that-1 *T*-1 wanted *PRO*-1 to leave home.*
REL: leave
ARG0: *PRO*-1
ARG1: home

Be careful to distinguish the above case from those of coordinated verb constructions, in which the relative clause should be annotated for each relation:

*The antiquities which-1 you have stolen *T*-1 and handed *T*-1 over to the antiquities mafia.*

REL: stolen
ARG0: you
ARG1: *T*-1
LINK-SLC: which-1 * The antiquities

Or, similarly. . .

*The antiquities which-1 you have stolen *T*-1 and handed *T*-1 over to the antiquities mafia.*
REL: [handed][over]
ARG0: you
ARG1: *T*-1
LINK-SLC: which-1 * The antiquities
ARG2: to the antiquities mafia

1.8.2 Pragmatic Coreference Link (Link-PCR)

Link-PCR is annotated through the use of either the '*' or '&' functions in Jubilee. Later, in post-processing, these functions are converted to the Link-PCR label.

Reduced Relative Annotation

A relative clause may be reduced when passive, resulting in the unique syntax of a reduced relative clause. For example, a passive relative clause construction such as '**The woman that was dressed in blue** walked past the house' can be reduced to '**The woman dressed in blue** walked past the house.' Because the verb in these cases is always passive, the TreeBank annotation of reduced relatives will include an object trace after the verb. However, unlike normal passive constructions, this trace will never be coindexed with the subject. Thus, when annotating the verb in a reduced relative construction, the annotator must firstly tag the trace as Arg 1 (just as one would do with a normal passive construction). Secondly, the annotator must select the appropriate subject NP node that the trace is associated with, and create the link between the object trace and the subject by clicking `Argument` on the Jubilee menu bar, followed by clicking `Functions`. From the options therein, select '&' (shortcut: Ctrl+Shift-7). At this point, the linked annotation should appear on the selected subject NP node.

TreeBank Annotation:

```
(S
  (NP-SBJ-1
    (DT This)
  (VP
    (VBZ is)
    (VP
      (VBN considered)
      (S
        (NP-SBJ
          (-NONE- *-1)
        (NP-PRD
          (NP
            (CD one)
          (PP
            (IN of)
            (NP
```

```
(NP
  (DT the)
  (JJS biggest)
  (NNS caches)
(VP
  (VBN seized)
  (NP
    (-NONE- *)
  (PP-LOC
    (IN in)
    (NP
      (DT the)
      (NN district)
```

PropBank Annotation:
REL: seized
ARG1: [NP -NONE- *] & [NP the biggest caches]
ARGM-LOC: [PP-LOC in the district]

Annotation of *PRO*

Many traces found in the TreeBank arise as a result of the movement of a constituent from its canonical position. Movement leaves a trace, represented by a * or a *T* in the TreeBank. *PRO*, on the other hand, does not arise as a result of movement. Rather, *PRO* arises where there is an underspecified, or unrealized subject of a verb. For example, the subject of the verb 'leave' in the phrase 'she tried to leave,' is not realized. However, the TreeBank will represent the unrealized subject of 'leave' with *PRO*:

*She-1 tried *PRO*-1 to leave*
REL: leave
ARG0: *PRO*-1

In cases like that of the example above, the *PRO* element is already coindexed with the fully realized subject because the *PRO* is positioned in a clause that is governed by the higher clause with the same subject, 'she.' Thus, the annotator need not add any additional links between *PRO* and the explicit subject. However, there are also cases in which *PRO* arises but it is not governed by a higher clause. In these cases, it is not coindexed with a fully realized subject. For example:

TreeBank annotation:

```
(NP-SBJ
   (NP
     (PRP it)
   (S
     (-NONE- *EXP*-1)
  (VP
   (VBZ is)
   (ADJP-PRD
     (JJS best)
   (S-1
     (NP-SBJ
       (-NONE- *PRO*)
```

```
      (VP
        (RB not)
        (TO to)
        (VP
          (VB use)
          (NP
            (PRP it)
          (S-CLR
            (NP-SBJ
              (-NONE- *PRO*)
            (VP
              (TO to)
              (VP
                (VB cut)
                (NP
                  (ADJP
                    (RB very)
                    (JJ hot)
                  (NN food)
```

PropBank annotation:
REL: cut
ARG0: [NP-SBJ -NONE- *PRO*]
ARG1: [NP very hot food]

When annotating *PRO* that is not indexed, if the annotator is certain that the subject is realized elsewhere in the instance, then a link should be created between *PRO* and the explicit reference. If the annotator is NOT absolutely certain that the explicit reference and *PRO* share the same referent, then the annotator should not create the link. Essentially, unless the relation is absolutely certain, we should err on the side of agreeing with the TreeBank annotation and its existing indices. In the cases where the annotator has decided with certainty that a link should be created between the *PRO* argument and a fully realized subject: first, annotate *PRO* with its appropriate argument, then select the node of the explicit subject mention associated with *PRO*, and finally, click `Argument` on the Jubilee menu bar, followed by clicking `Functions`. From the options therein, select '*' (shortcut: Ctrl+Shift-8). At this point, the linked annotation should appear on the currently selected node of the explicit referent.

TreeBank annotation:

```
(NP-SBJ
    (NP
      (NNP China)
      (POS 's)
    (NN income)
    (NNS taxes)
  (VP
    (VBD amounted)
    (PP-CLR
      (TO to)
      (NP
        (QP
          (RB approximately)
```

```
      (CD 180)
      (CD billion)
(, ,)
(S-ADV
  (NP-SBJ
    (-NONE- *PRO*)
  (VP
    (VBG accounting)
    (PP-CLR
      (IN for)
      (NP
        (NP
          (QP
            (RB about)
            (CD 6)
          (SYM \%)
        (PP
          (IN of)
          (NP
            (NP
              (DT the)
              (NN state)
              (POS 's)
            (JJ financial)
            (NNS revenues)
```

PropBank annotation:
REL: accounting
ARG0: [NP-SBJ -NONE- *PRO*] * [NP China's income taxes]
ARG1: [PP-CLR for about 6% of the state's financial revenues]

The goal of this annotation is to provide additional semantic information about the arguments of the verbs. In some cases, antecedents are not syntactic constituents, or have a different morphological form, as the possessive pronoun 'your' below illustrates; forego linking in these cases:

On the issue of abortion , Marshall Coleman wants to take away your right [*] to choose and give it to the politicians.
ARG0: [*PRO*]
REL: choose

Additionally, note that the null element should be linked to the highest possible node containing its referent without recursively annotating other arguments or the REL itself (i.e., creating arguments embedded within other arguments). If this is not possible, the link should be omitted.

1.8.3 Concatenation of multiple nodes into one argument

As a rule, annotations should be placed on the highest node possible encompassing the entirety of a constituent (e.g., the NP, or PP level in the TreeBank), and one tag should correspond to one node for every node within the appropriate annotation span. However, in some cases the rel is situated within an NP node, and the manner in which the TreeBank is laid out makes it

impossible to capture a whole constituent under one node. In these cases, two leaves of the tree may have to be concatenated together under a single argument label. For example:

TreeBank Annotation:

```
...
(ADJP
  (RB widely)
  (VBN covered)
  (NN skiing)
  (NN competition)
```

PropBank Annotation:
REL: covered
ARGM-MNR: [RB widely]
ARG1: [NN skiing] , [NN competition]

To concatenate two leaves into a single argument, first select the node of the first word of the constituent (e.g, 'skiing') and then click the appropriate button (e.g, 1) indicating which semantic role the argument is playing. Next, select the node of the second word of the constituent (e.g., 'competition') and navigate to `Argument` on the Jubilee menu bar, followed by clicking `Functions`. From the options therein, select ',' (shortcut: Ctrl+Shift-,). The resulting annotation will reflect the tree location of both nodes as part of a single argument (e.g., `9:0,8:0-ARG1`.

1.8.4 Special cases of topicalization

Topicalization occurs when a constituent is moved from its underlying syntactic position to an atypical position in order to draw attention to that constituent. Although the majority of cases of topicalization simply require annotation of a trace that is already indexed to the topicalized constituent, occasionally the topicalized constituent is repeated, often in the form of a pronoun, and the two constituents are not already indexed in the tree. For example, '911, that's the number you call in an emergency.' If the REL to be annotated is 'is,' the annotator would firstly have to annotate the pronoun 'that' as ARG-1, then provide a coreference link to the pronoun's referent, '911.' Performing this link works in an identical manner to that of creating coreference links for *PRO* seen in section 1.8.2. Thus, after selecting and annotating the pronoun in the argumental position, subsequently select the topicalized node and click `Argument` on the Jubilee menu bar, followed by clicking `Functions`. From the options therein, select '*' (shortcut: Ctrl+Shift-8). The linked annotation should appear in the TreeBank view and in the annotation view at the top of the screen. For example:

TreeBank Annotation:

```
(S
  (NP-SBJ
    (PRP I))
  (VP
    (VBD expected)
    (NP
      (PRP it))
    (PP-LOC
      (IN in)
```

```
(NP
  (NP
    (DT a)
    (NN country))
  (SBAR
    (WHNP-1
      (-NONE- 0))
    (S
      (NP-SBJ
        (PRP we))
      (VP
        (VBP love)
        (NP
          (-NONE- *T*-1))
        (, ,)
        (NP-TPC
          (PRP me)
          (CC and)
          (PRP you))))))))
```

PropBank annotation:
REL: love
ARG1: (NP (-NONE- *T*-1))
LINK-SLC: [WHNP-1 (-NONE- 0)] * [NP a country]
ARG0: [NP-SBJ we] *[NP-TPC me and you]

1.9 Special Cases: small clauses and sentential complements

This section is concerned with different types of clausal complements and modifiers. In the following sentence, the clause S-CLR has a trace in the subject position of 'asleep', which is coindexed with the subject of the verb 'fell,' 'I'.

I fell asleep on the floor.

TreeBank annotation:

```
S (NP-SBJ-1 I)
     (VP fell
         (S-CLR (NP-SBJ *-1)
                (ADJP-PRD asleep))
         (PP-LOC on
                 (NP the lobby floor))))
```

When annotating the verb 'fell', the small clause (marked as S-CLR above) is tagged as ArgM-PRD, and the Arg1 argument is the NP-SBJ 'I'. Note that although the empty category NP-SBJ *-1 is being coindexed with 'I', the trace is not the argument of 'fell', but rather is the subject of 'asleep'.

PropBank annotation:
REL: fell
ARG1: I

ARGM-PRD: [NP-SBJ *-1] asleep

Verbs like 'expect' are analyzed as having a clause as its argument (which corresponds to the event expected). In this case PropBank annotation follows TreeBank analysis of these sentences, where the clausal complement is being selected as Arg1:

John expected Mary to come.

PropBank Annotation
REL: expected
ARG0: John
ARG1: Mary to come

If such sentences are passivised, as shown below, then the Arg1 argument is the clausal complement of the verb. Parallel to ICH and RNR traces, we assume that the trace [*-1] is being 'reconstructed', so that the Arg1 in this case corresponds to the proposition 'Mary to come'. Again, similar to ICH and RNR traces, it is not necessary to annotate the dislocated portion (e.g., NP Mary-1) with another tag. The annotator can think of the dislocated portion as part of, and therefore 'covered' by the annotation of the sentential complement in which there is a trace co-indexed to the dislocated piece.

Mary-1 is expected [*]-1 to come
REL: expected
ARG1: [*-1 to come]

A similar analysis applies to verbs like 'seem' and 'appear', which are known as raising verbs. For example, the NP 'Everyone' is not the argument of the verb 'seems', but rather this sentence can be paraphrased as 'It seems that everyone dislikes Drew Barrymore.'

Everyone seems to dislike Drew Barrymore

TreeBank annotation:

```
(S (NP-SBJ-3 Everyone)
     (VP seems
         (S (NP-SBJ *-3)
            (VP to
                (VP dislike
                    (NP Drew Barrymore))))))
```

In PropBank annotation, the S clause is being annotated as the Arg1 argument and the dislocated 'everyone' is ignored:

REL: seems
ARG1: [NP-SBJ*-3 to dislike Drew Barrymore]

And, finally, another class of verbs which follows this analysis includes aspectual verbs like 'continue' and 'start', which take events as their arguments.

[New loans]-4 continue [-4] to slow.*
PropBank annotation:
REL: continue
ARG1: [*-4 to slow]

However, verbs such as these may require additional annotation IF the dislocated portion of the argument that has a co-indexed trace in the sentential complement is ANIMATE and AGENTIVE. Where this is so, it is annotated as ARG-0 in the dislocated position, and, as before, the entire S clause including the trace to the ARG-0 constituent is annotated as ARG-1:

[John]-1 began [-1] to read the book*
PropBank annotation:
REL: began
ARG0: John
ARG1: [*-1] to read the book

1.10 Handling common features of spoken data

Annotation of transcripts of spoken data tends to be more difficult than annotation of spoken material due to disfluencies, repetitions and asides that do not normally occur in written English. Annotation procedures for each of these types of challenges are addressed in the following sections.

1.10.1 Disfluencies and Edited Nodes

Speakers often begin to say one utterance, stop due to a variety of speech errors or pragmatic factors, and then resume the utterance. Sometimes the speaker resumes with a very similar utterance, other times the speaker resumes with what seems to be an entirely different utterance. The Treebank handles such disfluencies with the use of a separate node, generally labeled 'Edited,' such that the error portion of the utterance is separated from the remainder of the utterance within this node. If the relation is not within the edited node, simply ignore edited nodes and do not annotate them, regardless of whether or not they are within the span of annotation. If the relation is within the edited node, annotate in accordance with the normal span of annotation, but do not annotate anything past the cut off point of the utterance (this will often have a function tag in the TreeBank 'UNF,' indicating that the utterance is unfinished). In other words, treat everything before the speaker stops and switches the progression of the utterance as you normally would any relation; however, do not annotate anything beyond where the speaker stops the original utterance and then starts the repaired utterance. See figure 1.14 for an example of proper annotation.

This example brings to light several common challenges in annotating disfluencies. Notice in this example that the TreeBankers have assessed the portion of the utterance that is later repaired, and this 'error' portion is placed in an edited node. The argument of 'be' prior to the cut off is annotated as usual. However, the beginning of the relative clause construction is not annotated; this is omitted because there is no index on the relativizer 'that' as there normally would be, which is always linked to a trace somewhere in the rest of the utterance. As a result, there is no anchor within the clause for the relative clause construction, and it is therefore incomplete and shouldn't be annotated. Conversely, if the trace linked to the relativizer were present within the edited node, it would require normal annotation. Although the entire instance is not shown here, it is also notable that there is no way for the annotator to know precisely which sense to use because the utterance is incomplete. When possible, try to use context to make the best guess of what sense of the verb is appropriate. It is often helpful to consider the repaired utterance after the cut off because speakers sometimes continue with a very similar utterance. However, in cases such as this one, where there is little relevant context to this portion of the utterance and the repaired utterance seems very different from the cut off utterance, simply select the most frequent sense of the verb. The most frequent sense of the verb should be the .01 sense;

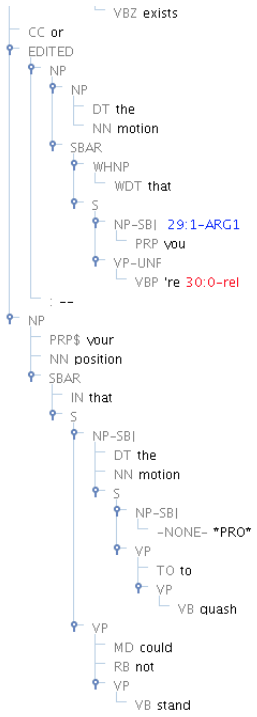

Figure 1.14: Correct annotation of disfluency in an edited node

however, most of the .01 senses in PropBank were established during annotation of the Wall Street Journal. Thus, when tackling a very different corpus that is a transcription of spoken data, it is possible that another sense of the verb will be particularly prevalent in that corpus. In these cases, annotators should use their best judgment of the patterns of that corpus to select what is the most likely sense of the verb in ambiguous cases.

1.10.2 Repetition: resumptive pronouns

Although resumptive pronouns are not used in written English, speakers sometimes insert a resumptive pronoun in spoken discourse. For example, 'The car that I drove IT yesterday'; 'Who is the girl that your friend said she's nice?'; 'Do you ever get suspicious that the Denver Post deletes blogs that they don't seem to fit?' These are accompanied by relative clause constructions. In these cases, the TreeBank should provide a node that encompasses both the resumptive pronoun and the trace linked to the relativizer. Annotate this node with the appropriate numbered or modifier label, but do not annotate the relative clause construction with the Link-SLC chain. Because the trace corresponding to the relativizer is embedded in a node with the resumptive pronoun, the relative clause construction should not be annotated (refer to 1.8.1 for further details on when to annotate link SLC and when to make exceptions). For example:

*G. could not write a paper that-1 I would not be able to tell that he wrote it *T*-1.*
ARG0: he
REL: wrote
ARG1: it *T*-1

1.10.3 Asides: PRN nodes

In both writing and perhaps somewhat more frequently in spoken discourse, speakers may insert an utterance that is not directly related to the main utterance in progress: 'It is the livestock sector, according to a new report by the United Nations Food and Agriculture Organization, that generates more greenhouse gases than any other industry.' In the preceding utterance, the 'according to' phrase would be encompassed in a node labeled 'PRN' (meaning parenthetical) in the TreeBank. Like 'Edited' nodes, 'PRN' nodes should not be annotated unless the relation is within the parenthetical node itself. Annotation within the PRN node should be restricted to the normal span, meaning that it will not extend beyond the PRN node. In the case of spoken discourse, such nodes are often used for embedded phrases such as 'I think,' 'You know,' or repair initiators such as 'I mean.'

Chapter 2

Light Verb Annotation

Light verb usages are those which are considered semantically bleached, thus they do not carry the specificity of meaning that the verb would carry outside of a light verb construction (Butt, 2003). For example, a 'heavy' version of the verb 'make' is used in the phrase 'She made a pie out of fresh cherries and refrigerated dough.' This usage reflects the normal semantic roles associated with the creation sense of 'make': agent, product and material. A light version of the verb 'make' is used in the phrase 'She made an offer to buy the company.' Unlike the first phrase, the verb 'make' does not specify the semantics of the event; rather, the action nominal complement or true predicate 'offer' specifies the event. For this reason, we can often rephrase light verb constructions with the verb counterpart of the action nominal complement without losing the meaning of the utterance: 'she offered to buy the company.' In addition to specifying the semantics of the event, the action nominal complement also projects the argument structure of the utterance. For example, the infinitival complement 'to buy the company' is a canonical type of argument for 'offer,' but it is not a canonical argument type for the verb 'make': '*She made to buy the company.' Because the verb in these cases is not the element that specifies the semantics of the event or projects the argument structure, we cannot treat light verbs in the same way that we treat 'heavy' verbs. Therefore, we have special annotation procedures for light verbs, outlined in the following sections.

2.1 Pass 1: Verb Pass

Common light verbs in English are 'make,' 'take,' 'have,' and 'do,' found in light verb constructions such as 'John made an inspection of the premises,' 'John took a walk to the store,' 'John had a drink of iced tea,' and 'John did an investigation of the crime.' The verb 'give' is often cited as a light verb in English as well, however, for the purposes of PropBank 'give' is not treated as a light verb because all light verb usages of 'give' maintain the canonical transfer semantics and ditransitive argument structure: 'She gave him thunderous applause.' Since PropBank commonly treats metaphorical extensions of one sense of the verb as equivalent to the concrete sense so long as the argument structure is similar, there is little reason to treat 'give' differently. Other light verbs, in contrast, do not maintain their canonical semantics or argument structure when used in light verb constructions. It is theoretically possible for many other verbs to be light in certain usages, such as 'She produced an alternation.' Whenever a verb seems to describe the event semantics less than the action nominal complement, it is likely a light verb construction.

When a light verb is encountered, the annotator should perform the following steps:

Step 1: roleset selection
Annotators should select the '.LV' roleset, available for all verbs, when the verb is found in

a light verb construction. In some cases, placeholder numbered rolesets are available for light verbs. Do not use this numbered roleset, always use the '.LV' roleset.

Step 2: annotation
Annotators should annotate only the subject of the light verb as Arg0, the direct object of the light verb containing the action nominal complement as ArgM-PRX (PRedicating eXpression), and the action nominal complement or true predicate itself as ArgM-PRR (PRedicating Relation). Unlike normal tag placement, the ArgM-PRR tag should be placed directly on the lexical level or leaf containing the action nominal complement. Figure 2.1 illustrates correct annotation of light verb constructions:

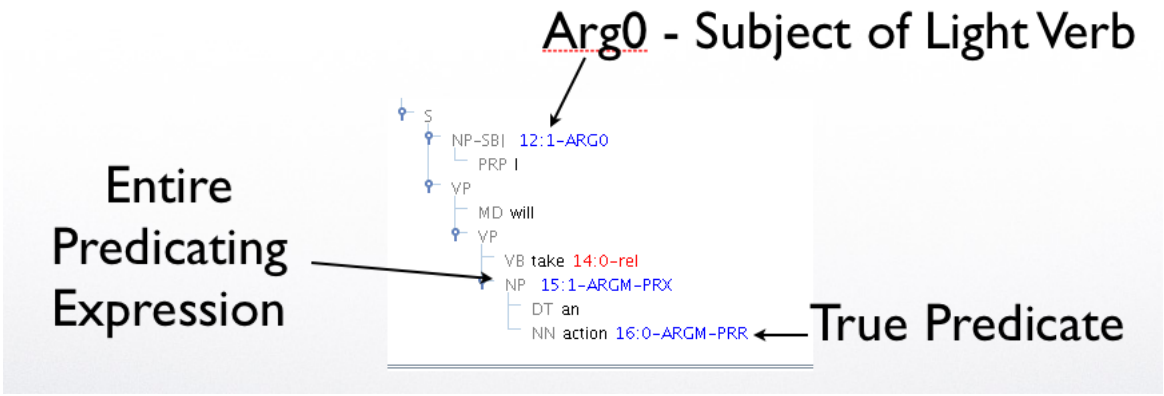

Figure 2.1: Correct annotation of a light verb construction

Annotate these three elements and ONLY these three elements. The first pass is strictly an identification pass meant to find and label light verbs as such. We do not annotate additional arguments because the verb itself is not the relation that is projecting the argument structure; thus it is not appropriate to annotate other arguments as if they were truly semantically related to the verb. Rather, the surrounding arguments are annotated during the second pass, when the action nominal complement is annotated, because it is the action nominal complement that projects the argument structure. However, where necessary, '*' and '&' links can be used to link the Arg0 or the ArgM-PRX to their referents. For example, Figure 2.2 shows an instance that requires the '&' link.

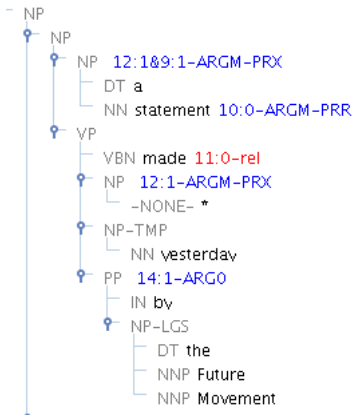

Figure 2.2: Light verb annotation including '&' link

However, Link-SLC should not be annotated, even in cases where a relativizer links the Arg0 or ArgM-PRX to its referent. Figure 2.3 shows a case of this type.

It is often difficult to decide if a certain usage is a light verb usage or not. Light verbs are

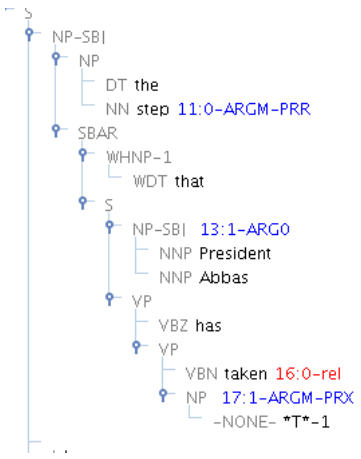

Figure 2.3: Light verb annotation, ignoring Link-SLC

thought to exist on a continuum ranging from the purely compositional meanings stemming from ordinary collocation of words (e.g. 'I tripped on the rug'), wherein every word maintains its full semantic value, to the entirely non-compositional meanings stemming from fixed idiomatic expressions (e.g. 'I tripped the light fantastic,' meaning 'to dance'). As a result, there are often fuzzy boundaries between heavy usages of verbs, light usages of verbs, and idiomatic usages of verbs. For the purposes of PropBank, annotators should be generous in their definition of light verb constructions and annotate accordingly when in doubt as to whether something is a light verb or not. In turn, cases where the annotator seems to have been too generous in this definition will be corrected in adjudication.

2.2 Pass 2: Noun Pass

The bulk of the annotation of light verb constructions will be performed during the second noun pass, wherein the action nominal complement or true predicate is the relation. Unlike ordinary noun annotation described in Chapter 3, annotation of nouns that are true predicates within a light verb construction requires the annotation of arguments within both the noun relation's span of annotation (sisters to the noun and sisters to the noun phrase), and the light verb's span of annotation (sisters to the verb and sisters to the verb phrase). In the case of light verb constructions, the syntactic arguments of both the light verb and the action nominal complement are annotated together because it is thought that both contribute to complex predication. After identifying the fact that the noun relation is a true predicate in a light verb construction, proceed with the following steps:

Step 1: roleset selection
Roleset selection proceeds exactly as it does for normal annotation of noun relations: select the appropriate numbered roleset according to the sense of the usage.

Step 2: annotation
Annotate direct arguments of the noun relation (sisters to the noun and sisters to the noun phrase) and the syntactic arguments of the light verb (sisters to the verb and sisters to the verb phrase) in accordance with the argument structure outlined in the selected roleset. In addition, annotate the light verb itself using the ArgM-LVB tag, place this tag directly on the leaf node of the light verb. For example:

Yesterday, Mary made an accusation of duplicity against John because she was enraged with jealousy.
ARGM-TMP: Yesterday

ARG0: Mary
ARGM-LVB: made
REL: accusation
ARG2: of duplicity
ARG1: against John
ARGM-CAU: because she was enraged with jealousy.

Thus, all arguments of the complex predicate (in this case make+accusation) are annotated in accordance with the true predicate's argument structure, outlined in the noun relation's roleset.

Chapter 3

Noun Annotation Instructions

Nouns can also be relations in the same way that verbs are often relations. Nominalizations and eventive nouns are especially likely to have an argument structure similar to that of a verb. For example, 'The troops' destruction of the city,' can be thought of as the nominal counterpart of the clause 'The troops destroyed the city.' Thus, PropBank treats certain nominalizations and eventive nouns as relations to be annotated in the same manner as verb annotations: 1) select the appropriate roleset created specifically for the noun relation, 2) annotate numbered arguments in accordance with this roleset, and 3) annotate modifier arguments with the appropriate ArgM tags. However, there are a few differences to be aware of when annotating nouns, which are outlined in the following sections.

3.1 Span of Annotation

The span of annotation for noun relations mirrors that of verb relations. Instead of annotating the sisters to the verb and sisters to the verb phrase, annotation is required for the sisters to the noun and sisters to the noun phrase that encompasses the noun relation. Figure 3.1 illustrates the noun relation's span of annotation. Also like the verb span of annotation wherein several VP nodes may encompass a verb, several NP nodes may encompass the noun relation, and arguments of the noun relation to be tagged can be sisters to each of these NP nodes. In the case of verbs, the annotator can use the position of the clause boundary to determine where the span of annotation is delimited. However, in the case of nouns, the boundary of the span is marked by another NP (or occasionally NOM) node. Annotators must determine which NP node is the stopping point by finding which is a sister to the verb. Annotators cannot annotate sisters to the highest NP node that is a sister to the verb because this will entail annotating sisters to the verb itself. This is illustrated in Figure 3.2. The long arrows indicate valid arguments of the noun relation, while the top T indicates the sister to the highest NP node, which is a sister to the verb 'is,' thus, annotators should consider this NP node as the stopping point for annotation.

3.2 Annotation of Numbered Arguments

Numbered arguments for noun relations are outlined in the corresponding noun frame file and rolesets therein. These should be selected in Jubilee in the same manner as verb rolesets. It is important to keep in mind that the type and order of constituents is highly variable for noun relations. For example, both agents and patients can appear before the relation as a possessor or noun modifier, or in a prepositional argument after the relation.

Channel Nine's broadcast of the nightly news was praised for its quality.

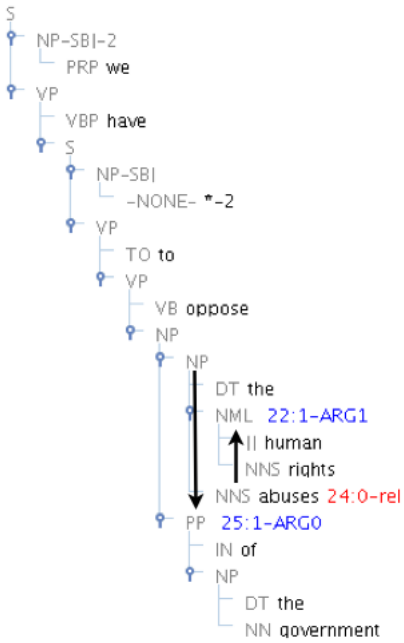

Figure 3.1: Noun span of annotation: the arrow to the right points to the sister of the noun, and the arrow to the left points to the sister of the noun phrase encompassing the noun relation

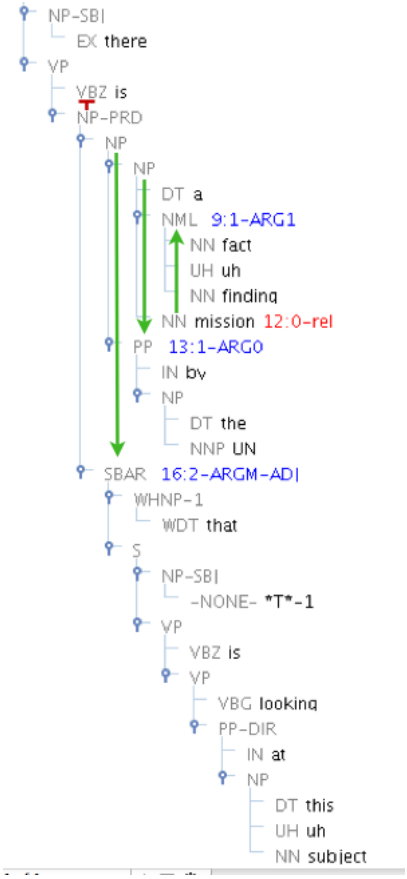

Figure 3.2: Noun span of annotation: the arrows point to arguments of the noun relation that should be tagged, while the highest NP node with the 'T' above it indicates the delimiting node of the span

ARG0: Channel Nine's
REL: broadcast
ARG1: of the nightly news

The nightly news broadcast of Channel Nine was praised for its quality.
ARG1: nightly news
REL: broadcast
ARG0: of Channel Nine

As a result of this variability, it is especially important to think of the primary goal of PropBank when annotating noun relations: assign the appropriate semantic tag across various syntactic realizations of the argument.

3.3 Annotation of Modifiers

Modifier arguments are identical for nouns with one exception, the ARGM-ADJ tag described below. That which is often expressed as adverbs in the case of verb relations is expressed with adjectives for noun relations:

Was your personal experience of having been on the cruise pleasurable?
ARG0: your
ARGM-MNR: personal
REL: experience
ARG1: of having been on the cruise

In the above example, one can think of 'personal' as equivalent to 'personally' in the clause counterpart of the phrase: 'You personally experienced having been on the cruise...' If it is helpful, try to rephrase the instance in this manner to decide what type of argument it would be when accompanying a verb, and it will be the same type of modifier when accompanying the noun relation.

3.3.1 Adjectival modifiers (ADJ)

Instead of the modifier tag 'ADV,' which should never be used for noun relations, the modifier tag ArgM-ADJ is used to label arguments that cannot be appropriately labeled with any other ArgM tag. As with the argM-ADV tag, the ArgM-ADJ tag should only be used as a last resort, when no other argument label can possibly fit. For example:

The mayor's shocking abuse of public funds outraged citizens.
ARG0: The mayor's
ARGM-ADJ: shocking
REL: abuse
ARG1: of public funds

In the above example, the manner of the abuse is not shocking, '?he shockingly abused public funds;' rather, the entire event is perceived as shocking by outsiders. Its verb counterpart would be something akin to 'Shockingly, the mayor abused public funds,' and would be annotated as ARGM-ADV for lack of a better tag. Similarly, modifiers such as this must be tagged as ARGM-ADJ for lack of a more specific tag. SBAR modifiers, as seen in Figure 3.2 should also be tagged as ARGM-ADJ.

3.4 Exceptions to annotation: determiners and other noun relations

As with verb annotation, the articles 'a,' 'an,' 'the' should not be annotated when they are in the span of annotation for a noun relation, nor should the majority of other determiners ('this,' 'that,' 'some,' 'any,' etc.), labeled 'DT' in the treebank. The greatest exception to this is, of course, possessive determiners which frequently correspond to numbered arguments. Additionally, certain determiners such as 'first' and 'final' can be annotated as ARGM-TMP when they denote the time or frequency of the event. Finally, negative determiners 'no,' and 'neither' should be tagged as ARGM-NEG.

Just as we do not annotate other coordinated verbs that fall within the span of annotation of a given verb relation, we do not annotate other coordinated eventive nouns or nominalizations that fall within the span of annotation of a given noun relation. For example:

Mary's investigation and eventual condemnation of the local government made the news.
ARG0: Mary's
ARGM-TMP: eventual
REL: condemnation
ARG1: of the mayor

Outside of these exceptions, all other arguments within the span of annotation should be annotated.

Chapter 4

Jubilee Annotation Tool Technical Guidelines

Jubilee is a PropBank instance editor developed at the University of Colorado at Boulder. For each verb or noun predicate in every tree (representing the phrase structure of the corresponding sentence), we create a PropBank instance that consists of the sense ID of the predicate (e.g., `run.02`) and its arguments labeled with the semantic roles (e.g., `ARG0`, `ARG1`). If more than one verb occurs in a tree, several PropBank instances can be generated from one tree. The argument structure of each predicate is outlined in the PropBank frameset or roleset of that predicate, an xml file containing semantic and syntactic information about each sense of a given predicate lemma that has been encountered thus far in PropBank annotation. Table 4 shows an example of a PropBank instance associated with a verb, 'run'.

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

Table 4.1: An example of PropBank instance associated with 'run'

With Jubilee, the entire annotation procedure can be done using one tool that simultaneously provides rich, graphical syntactic information as well as comprehensive semantic information from the frame file.

Jubilee 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.[1] It is light enough to run as an X11 application. This aspect is important because both TreeBank and PropBank files are usually stored in a server, and annotators and adjudicators are to work on the tasks remotely (via SSH), using their local machines. One of the biggest advantages of using Jubilee is that it runs on many different languages; in fact, it has been used for PropBank projects in Arabic (Diab et al., 2008), Chinese (Xue and Palmer, 2009), English (Palmer et al., 2005) and has also been tested on Hindi and Korean (Han et al., 2002).

This report details how to setup and run Jubilee in normal and gold modes. The normal (annotation) and gold (adjudication) mode is meant to be run by annotators and adjudicators, respectively. In normal mode, annotators are allowed to see and open only tasks that have been claimed by themselves or those that have been claimed by one other annotator (when the maximum number of annotators allowed for each task is two). In gold mode, adjudicators are

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

allowed to see and open all tasks that have undergone at least single-annotation. Adjudicators can then either choose appropriate annotations or modify annotations if necessary to set the gold standard annotation. Although there are two different modes, the interfaces are very similar, so learning one mode effectively teaches the other.

4.0.1 Getting started

4.0.2 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.

4.0.3 Download and install Jubilee

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

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

This command will create two folders: `resource` and `system`. The `resource` folder contains resources required for the PropBank annotation. There are four sub-folders under `resource` folder: `annotation`, `frameset`, `task` and `treebank`. The `annotation` folder is initially empty, but will be filled with annotation files. The `task` folder contains a sample task file, `AFGP.task`, including four unannotated instances. The `frameset` and `treebank` folders contain frameset and TreeBank files used for the task.

The `system` folder contains configuration files required to run Jubilee. There are three kinds of configuration files in the folder: a language-argument file, a function-argument file and a project-path file. The argument file contains the language-specific semantic role labels and their keyboard shortcuts used in Jubilee (see Section **??** for more details). The naming convention of the language-argument file is as follows.

```
$LANG    ::= arabic | chinese | english
FILENAME ::= $LANG.args (e.g., english.args)
```

The function-argument file (`function.args`) contains operators and their keyboard shortcuts used for PropBank annotations (see Section **??** for more details). Finally the project-path file contains directory paths of all resource folders (`annotation`, `frameset`, `task` and `treebank`). When you start a new PropBank project, you need to create a new project-path file that contains directory paths of appropriate resource folders used for the project. The resource folders can be under any location that the project-path file points to; they do not have to be under `resource` folder. The naming convention of the project-path file is as follows.

```
$LANG    ::= arabic | chinese | english
$PROJECT ::= name of the project
FILENAME ::= $LANG.$PROJECT.path (e.g., english.sample.path)
```

4.0.4 Run Jubilee

To run Jubilee, type the following command on the terminal.

```
Usage: java -jar jubilee-version.jar <user ID> [max annotators = 2] [skip = 0]
```

`<user ID>` is a required parameter that indicates the user ID. Jubilee uses this user ID to distinguish tasks annotated by different users. If you use '`gold`' as the user ID, Jubilee will run in gold mode. The `[max annotators]` portion indicates the maximum number of annotators allowed to annotate each task (the default value is 2). The `[skip]` flag is used only for gold mode. If `skip` is equal to 1, Jubilee skips instances whose annotations (done by different annotators) are the same. This option is useful when annotators are well-trained and seasoned, so that annotations agreed upon by the annotators are considered correct, and therefore do not need to be checked by the adjudicator. If `skip` is equal to 0, Jubilee does not skip such instances. This option is useful when many annotators are new or annotation procedures have undergone changes; thus, annotations agreed upon by two annotators may still be incorrect and require modification by the adjudicator. The default value for the `[skip]` flag is 0.

4.0.5 Jubilee in normal mode

4.0.6 Launch Jubilee for annotators

Annotators (not adjudicators) are expected to run Jubilee in normal mode. In normal mode, annotators are allowed to view and edit only annotations done by themselves. To run Jubilee in normal mode, type the following command in the terminal.

```
java -jar jubilee-version.jar choijd
```

This will launch Jubilee for a user `choijd` in normal mode, allowing two annotators for each task. If you want to allow three annotators for each task, type the following command instead.

```
java -jar jubilee-version.jar choijd 3
```

When you launch Jubilee, you will see an open-dialog (Fig. 4.1). There are three components in the dialog. The combo-box at the top shows a list of project-path files in the `system` folder (Section 4.0.3). Once you select a project (e.g., `english.sample`), both `New Tasks` and `My Tasks` lists will be updated. `New Task` shows a list of task files that have either not been claimed, or claimed by only one other annotator (when there are two annotators allowed for each task). `My Tasks` shows a list of task files claimed by the current user (e.g., `choijd`). In any case, you can choose only one task at a time.

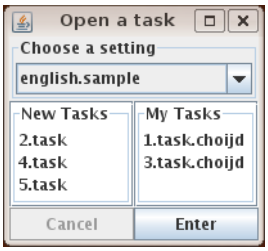

Figure 4.1: Open-dialog

Once you choose a task and click `Enter` button, the Jubilee main window will be prompted (Fig 4.2).

There are three kinds of views available in the main window: treebank view, frameset view and argument view. The following sections explain what functions are there in each view.

4.0.7 TreeBank view

By default, the treebank view shows the first tree in the selected task file. 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 where appropriate. The verb predicate is marked with a special tag '`rel`'. You can navigate through different trees in the task by clicking `[Prev]` and `[Next]` buttons (shortcut: `,` and `.`) or `[Jump]` combo-box for quick jump (`Ctrl+J`).

The annotator text-box on the right shows your ID (e.g., `choijd`) indicating the user annotating this instance. The sentence text-box at the bottom shows the raw sentence of the tree. To view the tree in text format, click `[TreeBank - View Tree in Text]` on the menu (`Ctrl+T`).

4.0.8 Frameset view

The frameset view displays and allows the annotator to choose from one or more senses of a predicate. The predicate text-box on the left shows the lemma of the predicate associated with

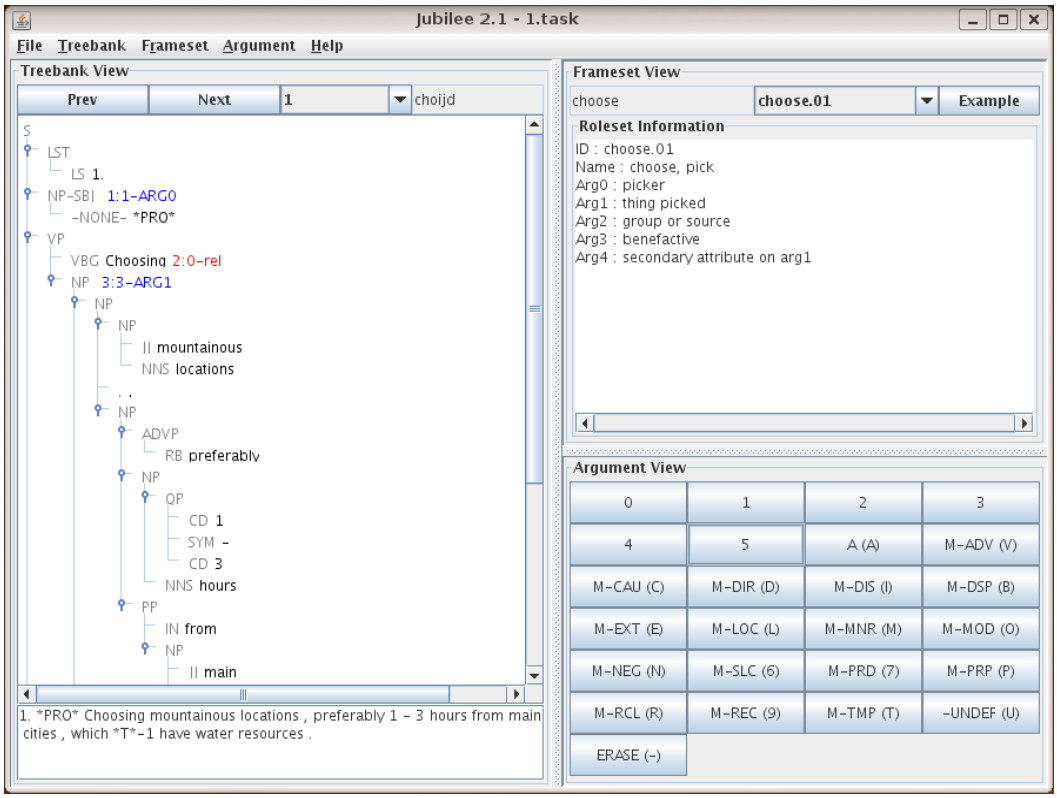

Figure 4.2: Jubilee main window

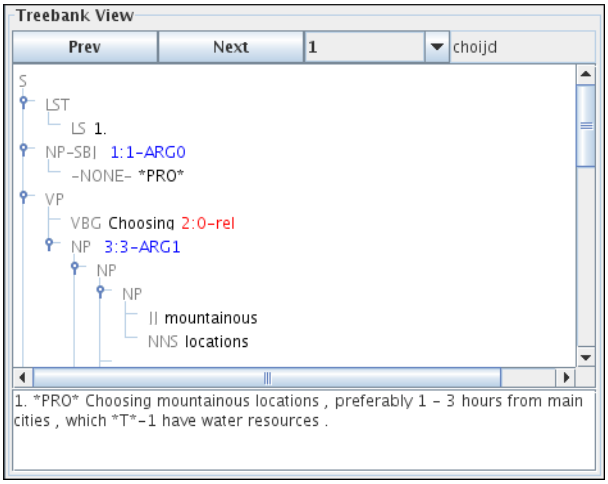

Figure 4.3: TreeBank view

the selected roleset[2] in the roleset combo-box. Initially, the predicate lemma followed by `.XX` is displayed here to indicate that no sense has been chosen yet (e.g., `choose.XX`). Annotators can peruse the available numbered senses of a predicate by clicking on the roleset combo-box, or they can move through the available rolesets sequentially by using the shortcut `]` to move forward, or `[` to move back to lower-numbered senses. As a roleset is selected, the argument structure and a short definition of that sense, which are extracted from the corresponding frameset file (e.g., `choose.xml`), appear in the roleset information pane. To view annotation examples of the currently selected roleset, click `[Example]` button (`Ctrl`+E). To view the arguments annotated in the current tree, click `[Frameset - View Arguments]` on the menu (`Ctrl`+W).

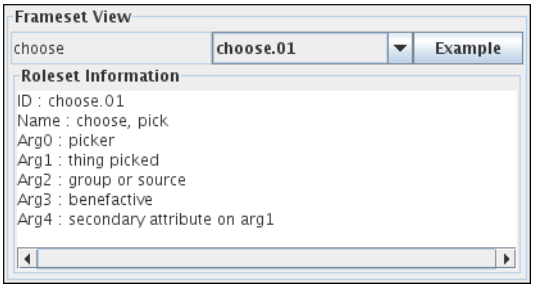

Figure 4.4: Frameset view

[2]The term 'roleset' is interchangeable with a term 'frameset' depending on languages.

4.0.9 Argument view

The argument view contains buttons representing each of the PropBank argument labels (Fig. 4.6). 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]`). Alternatively, you could use a keyboard shortcut to insert the argument (e.g., `0`). All keyboard shortcuts can be found from the menu, `[Argument - Arguments]`. When you click the button (or use the shortcut), an argument label (e.g., `ARG0`) is annotated on the selected node, along with its relative location in the tree (e.g., `1:1`). The first number 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 word-token. For example, `NP-SBJ` node in Fig. 4.3 is annotated as `1:1-ARG0`. This indicates that the node has semantic role of `ARG0` and its location in the tree is `1:1` such that the 1st word-token, `*PRO*`, is the closest token-level node and `NP-SBJ` is `1` phrase level higher than the `*PRO*` node.

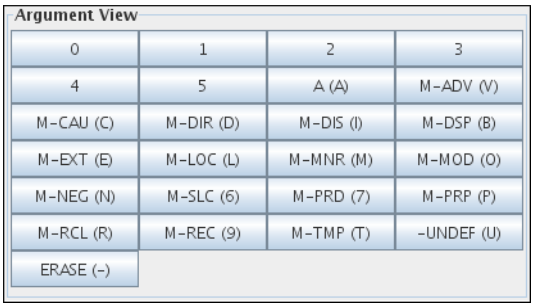

Figure 4.5: Argument view

The `[Erase]` button is used to remove the annotation assigned to the node. `[-UNDEF]` button is used to assign an argument that is not defined in the predicate-argument stucture of the currently selected roleset. For example, if the selected roleset has only `ARG0` and `ARG1` in its argument structure, but the annotator found a node in the tree that should be `ARG2`, `ARG-UNDEF` tag is used to indicate potential changes in the corresponding frameset file.

4.0.10 Annotate traces, concatenations and links

Several operators are used to annotate certain traces, and to perform concatenations and links. In the absence of TreeBank coindexing between a null element and its fully realized referent, annotators can provide semantic information about the null element serving as an argument by manually linking it to a fully realized referent in the instance by using '`*`' . For example,

TreeBank annotation:

```
...
  (NP-SBJ
    (NP
      (NNP China)
      (POS 's)
    (NN income)
    (NNS taxes)
  (VP
    (VBD amounted)
    (PP-CLR
      (TO to)
      (NP
```

```
      (QP
        (RB approximately)
        (CD 180)
        (CD billion)
    (, ,)
    (S-ADV
      (NP-SBJ
        (-NONE- *PRO*)
      (VP
        (VBG accounting)
        (PP-CLR
          (IN for)
          (NP
            (NP
              (QP
                (RB about)
                (CD 6)
              (SYM \%)
            (PP
              (IN of)
              (NP
                (NP
                  (DT the)
                  (NN state)
                  (POS 's)
                (JJ financial)
                (NNS revenues)
```

PropBank annotation:
Rel: accounting
Arg0: [NP-SBJ -NONE- *PRO*] * [NP-SBJ China's income taxes]
Arg1: [PP-CLR for about 6% of the state's financial revenues]

This operator is also used to create a semantic link between relativizers and their referents, which are never coindexed in the TreeBank:

TreeBank annotation:
(NP **(NP answers)**
(SBAR **(WHNP-6 that)**
(S (NP-SBJ-3 we)
(VP 'd
(VP like
(S (NP-SBJ *-3)
(VP to
(VP have
(NP *T*-6))))))))))

PropBank annotation:
Arg1: [NP *T*-6]
ArgM-LINK-SLC: [WHNP-6 that] * [NP answers]
Rel: have

Arg0: [NP-SBJ *-3]

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:

TreeBank Annotation:

```
(S
  (NP-SBJ-1
    (DT This)
  (VP
    (VBZ is)
    (VP
      (VBN considered)
      (S
        (NP-SBJ
          (-NONE- *-1)
        (NP-PRD
          (NP
            (CD one)
          (PP
            (IN of)
            (NP
              \textbf{(NP
                (DT the)
                (JJS biggest)
                (NNS caches)}
              (VP
                (VBN seized)
                \textbf{(NP
                  (-NONE- *)}
                (PP-LOC
                  (IN in)
                  (NP
                    (DT the)
                    (NN district)
```

PropBank Annotation:
Rel: seized
Arg1: [NP -NONE- *] & [NP the biggest caches]
ArgM-LOC: [PP-LOC in the district]

Additionally, in the relatively rare cases wherein an argument is discontinuous such that it cannot be captured in the annotation of one node, ',' can be used to concatenate more than one node into a single argument:

TreeBank Annotation:

```
...
(ADJP
  (RB widely)
```

```
(VBN covered)
(NN skiing)}
(NN competition)}
```

PropBank Annotation:
Rel: covered
ArgM-MNR: [RB widely]
Arg1: [NN skiing] , [NN competition]

The following shows how to provide the semantic * link between a null element and its referent. Other operators work in a similar way.

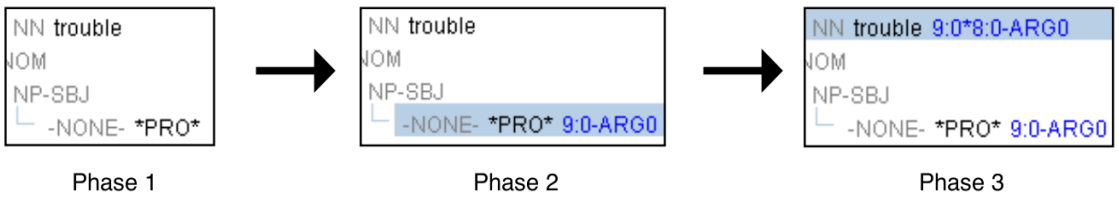

Figure 4.6: Argument view

1. Assume that `*PRO*` is `ARG0` and `trouble` is the node it needs to be linked to.
2. We first need to annotate `*PRO*` as `ARG0`. Select the `*PRO*` node in the treebank view and click either `[0]` button in the argument view or '`0`' on the keyboard. This inserts the argument label (`ARG0`) with its relative location in the tree (`9:0`) to the selected node (`*PRO*`).
3. Next, select the `trouble` node and type `Ctrl+Shift+8`. Jubilee remembers the last annotation you inserted (`9:0-ARG0`) and adds the annotation with a * link to the currently selected node. Hence, the final annotation of `trouble` becomes `9:0*8:0-ARG0` where `8:0` is the location of the `trouble` node.

4.0.11 Annotate verb particle constructions

Some English verbs use particles to form their predicate lemmas (e.g., 'run up,' as in 'run up a bill'). For such verb particle constructions, the particle is concatenated with the verb to form a single predicate lemma, annotated with the '`rel`' 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; for example, `9:0,8:0-rel`. Additionally, if for some reason the annotator accidentally erases the rel, it can be annotated again using the same command, `Ctrl+Shift+/`.

4.0.12 Save annotations

All annotations are saved automatically as you move on to a different tree. Notice that Jubilee may not save the very last tree you worked on; in this case, you can manually save the annotations by clicking `[File - Save]` on the menu (`Ctrl+S`). The annotation file is saved with a filename `*.task.userID` and can be found in the annotation folder (Section 4.0.3). For example, if the currently selected task is `1.task` and your user ID is `choijd`, the annotations will be saved to a file `1.task.choijd`. If you want to save the annotations to some place other than the default location, click `[File - Save As]` on the menu.

4.0.13 Jubilee in gold mode

Adjudicators are expected to run Jubilee in gold mode. In gold mode, adjudicators are allowed to view and edit all annotations. To run Jubilee in gold mode, type the following command in the terminal.

```
java -jar jubilee-version.jar gold
```

This will launch Jubilee in the default gold mode, which accommodates and displays two annotations for each instance. Notice that the only difference between running in normal and gold mode is the use of a user ID `gold`. In Jubilee, `gold` is a reserved ID only for adjudicators. If you want to view three annotators' annotations for each instance, type the following command instead.

```
java -jar jubilee-version.jar gold 3
```

This will launch Jubilee in gold mode, viewing three annotations for each instance. If you want to skip the instances that all annotators agreed on, type the following command.

```
java -jar jubilee-version.jar gold 2 1
```

This will launch Jubilee in gold mode, viewing two annotations for each instance and skipping instances for which all annotations are the same.

When you launch Jubilee, you will see the same open-dialog as you saw in Fig. 4.1. The `[New Tasks]` shows a list of task files that have not been adjudicated and the `[My Tasks]` shows a list of task files have been adjudicated. In gold mode, however, Jubilee does not allow opening tasks in `[New Tasks]` unless one or more annotation files created for the task already exists. If you try to open a task that has not been annotated by any annotator, it gives you an error message (Fig. 4.7). This prevents adjudicators from claiming tasks that have not yet been annotated.

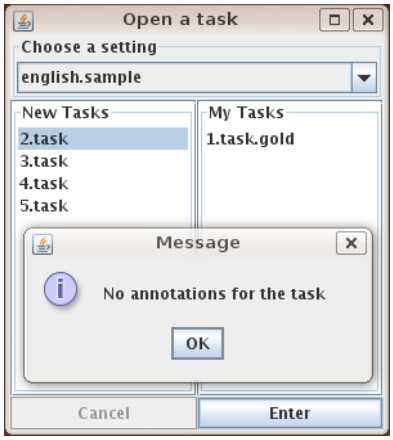

Figure 4.7: Open-dialog in gold mode

Once you launch Jubilee in gold mode, you will see almost the same main window as in Fig. 4.2, except now you have a list showing all annotations for the current instance at the top of the treebank view (Fig 4.8). The first line shows the annotation done by adjudicator, and the following lines show annotations done by different annotators. By default, the annotation from the first annotator is chosen as gold, but it can be changed by clicking the other annotation. To modify the annotation in cases where neither is perfectly correct, the adjudicator can simply manipulate the arguments as you would in the normal annotation mode.

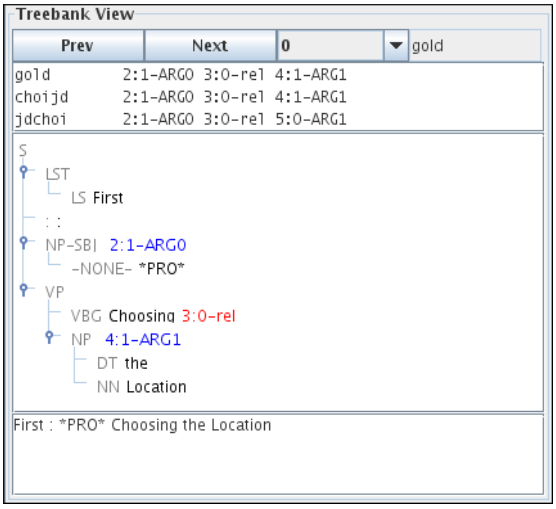

Figure 4.8: The treebank view in gold mode

When you save the adjudications, Jubilee creates an adjudication file called `*.task.gold` (e.g., `1.task.gold`) in the annotation folder.

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 (COLING-ACL'98)*.

Miriam Butt. 2003. The light verb jungle. In C. Bowern G. Aygen and C. Quinn, editors, *Papers from the GSAS/Dudley House Workshop on Light Verbs.*

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)*.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(1):547–619.

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)*.

Beth Levin and M. Rappaport Hovav. 1995. *Unaccusativity: At the Syntax-Lexical Semantics Interface; Linguistic Inquiry Monograph*. MIT Press.

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.

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.

Chapter 5

Cornerstone Frame Creation Tool Technical Guidelines

5.1 Introduction

Cornerstone is a PropBank frameset editor developed at the University of Colorado at Boulder. 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`). Adjudicators can add senses to the frameset file or create entirely new framesets 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'). All frameset files are written in Extensible Markup Language (XML) format. XML provides a useful, hierarchical format that is suited to the project.

Each sense of the predicate, alternately referred to as a '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.

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

[1] The language-dependent distinction between roleset and frameset is elaborated in Section 5.4.1

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.

[2]The current version of Cornerstone also runs on JDK 5.0 but running on JDK6.0 is preferable for the future updates.

5.2 Getting started

5.2.1 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.

5.2.2 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.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. All system files are in text format, so you could update 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 5.2.3 for more details).

5.2.3 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 LANG USER_ID
```

`LANG` 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 that indicates a path to the directory. This configuration file is 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.

5.3 Cornerstone in multi-lemma mode

5.3.1 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. 5.1 shows what it looks when you open the frameset file, `run.xml`.

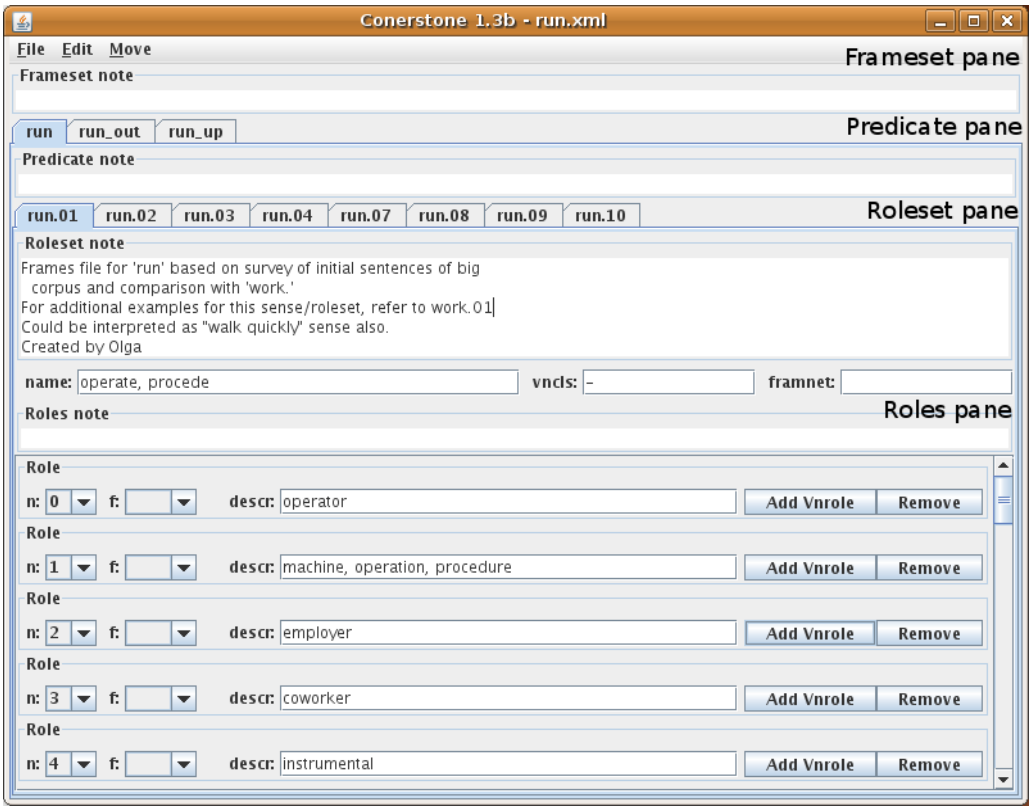

Figure 5.1: Open `run.xml` frameset file

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 (Baker et al., 1998).[3] In addition, the roleset pane contains three attribute fields (`name`, `vncls`, and `framnet`) and a roles pane. The `name` attribute shows 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 5.1).

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

Table 5.1: 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 5.2 in page 79 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 5.3 for a complete list of VerbNet thematic roles). As mentioned in Section 5.1, 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

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

the VerbNet class or thematic role should be omitted.

5.3.2 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 5.2 shows what it looks when you create a frameset file `temp.xml`.

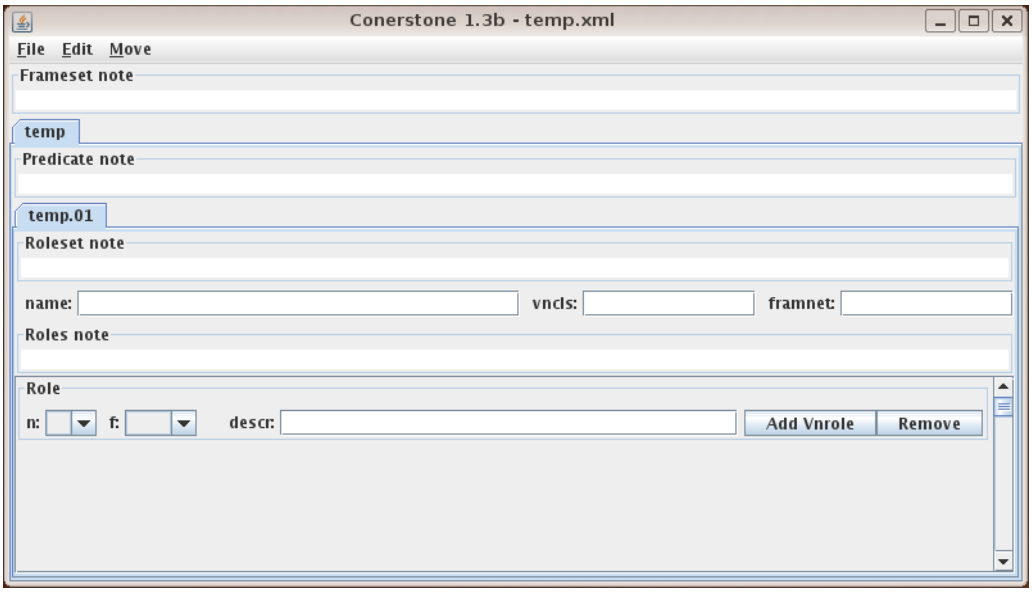

Figure 5.2: Create `temp.xml` frameset file

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').

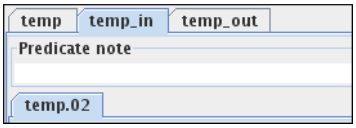

Figure 5.3: Add predicate lemma 'temp in' and 'temp out'

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

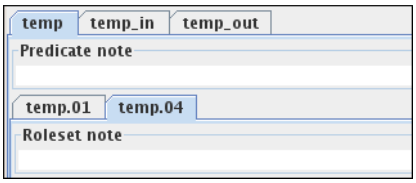

Figure 5.4: Add a roleset `temp.04` to a predicate 'temp'

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 5.2).[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 (Figure 5.5).

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 5.2: List of function tags in English

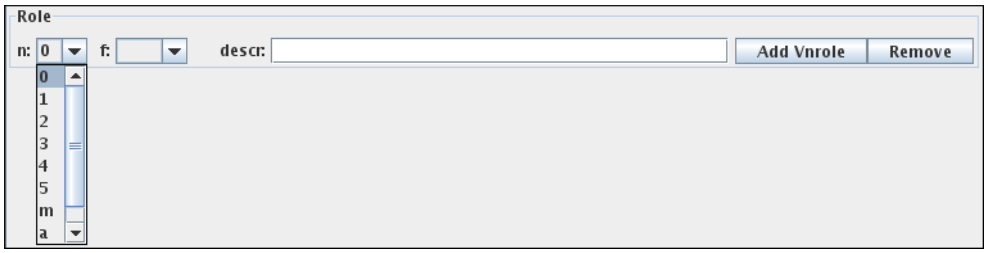

Figure 5.5: Edit and remove role

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

[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 5.3). To remove a `vnrole`, click `[remove]` button on the right (Fig. 5.6).

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 5.3: List of VerbNet thematic roles

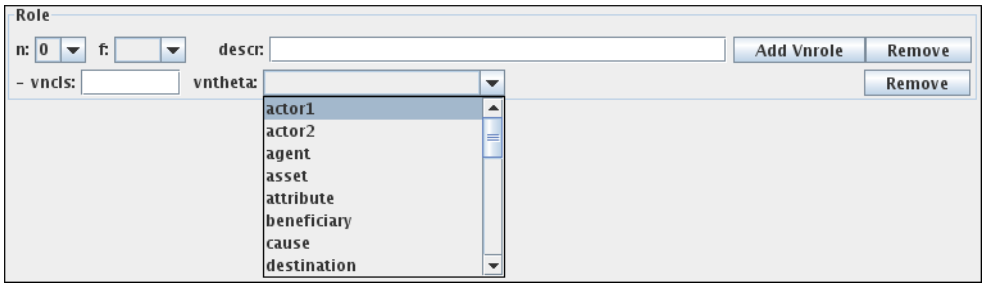

Figure 5.6: Edit and remove `vnrole` (VerbNet role)

5.3.3 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 5.7 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+−`).

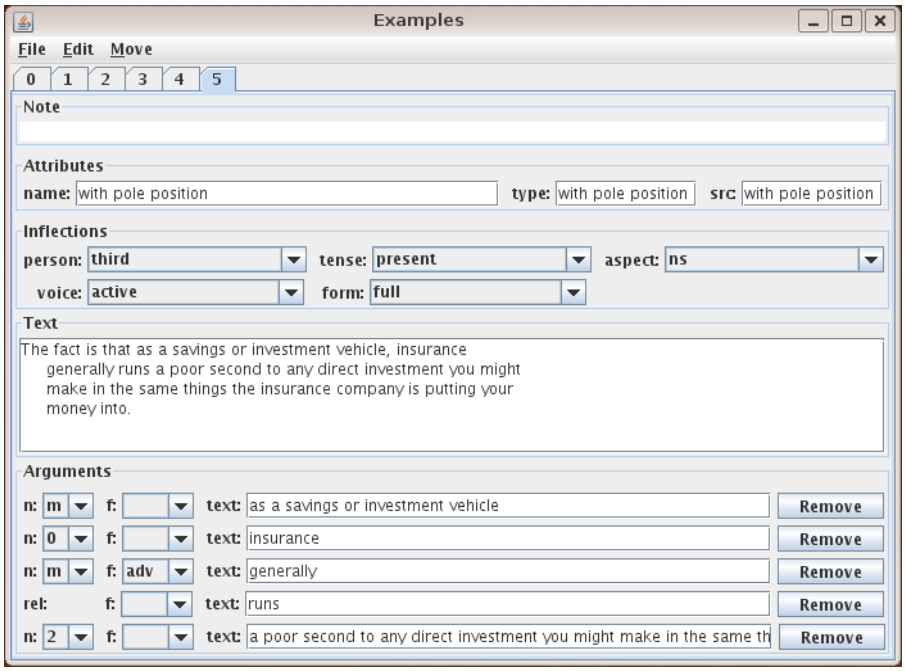

Figure 5.7: Example frame for `run.02`

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 5.4.

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 5.4: 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 relativizer (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 relativizer 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.

5.3.4 Save a frameset file

Your work will not be saved unless you assign the job to Cornerstone. 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.

5.4 Cornerstone in uni-lemma mode

5.4.1 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. 5.8 shows what it looks when you open an Arabic frameset file `HAfaZ.xml`.

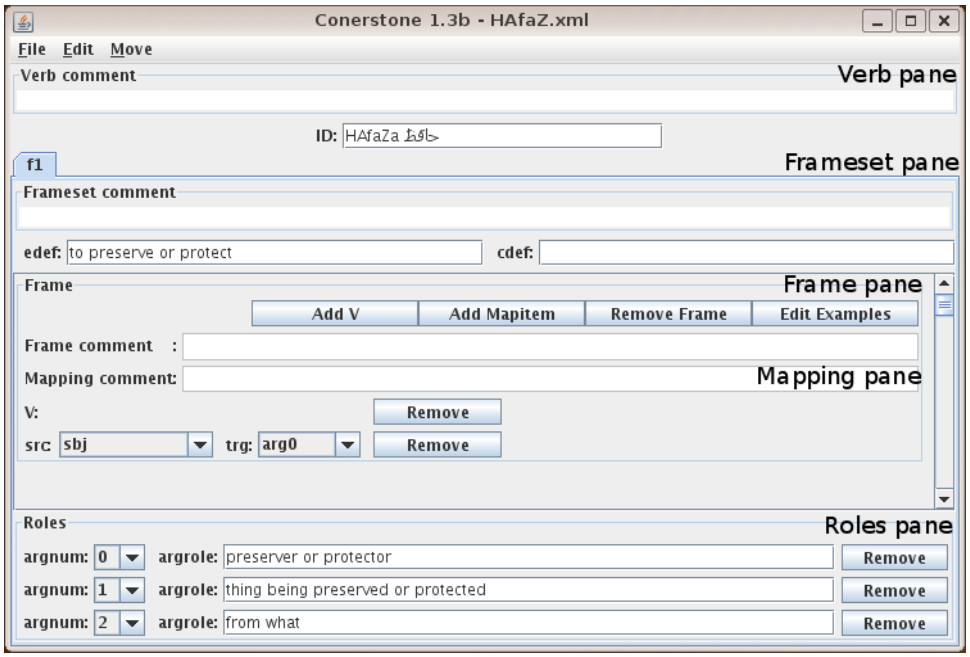

Figure 5.8: Open `HAfaZ.xml` frameset file

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 theTreeBank (Marcus et al., 1993), in which case, these mappings can be used for automatic extraction of semantic arguments from their syntactic arguments. Table 5.5 (page 86) 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 5.1 (page 76).

5.4.2 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. 5.9 shows what it looks when you create a frameset `temp.xml`.

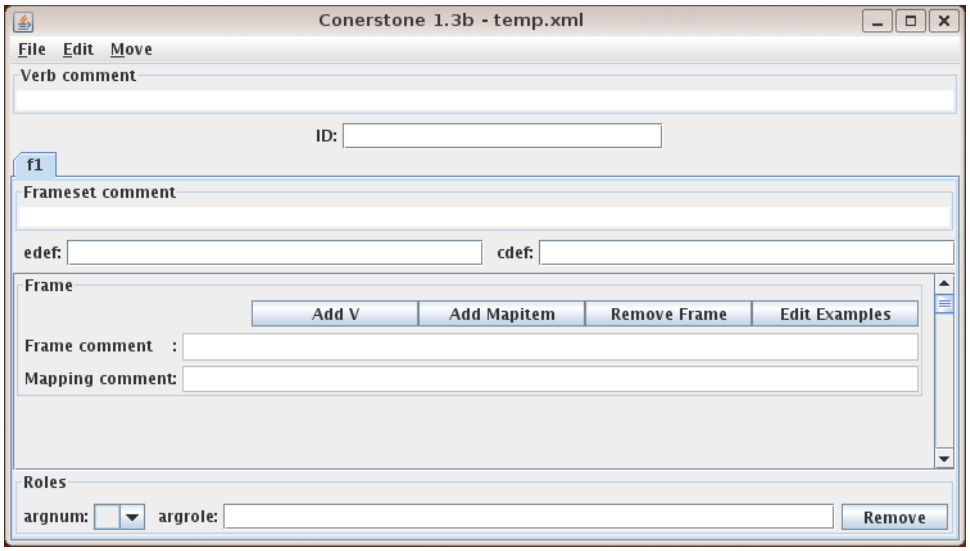

Figure 5.9: Create `temp.xml` frameset file

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 5.10: 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. 5.11).

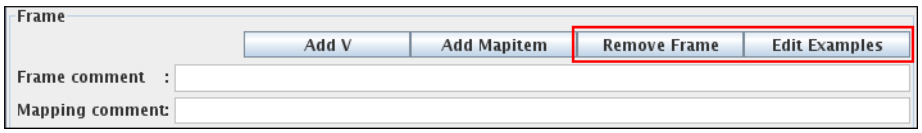

Figure 5.11: Add Remove a frame

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 5.5). Similarly, when you click `trg` combo-box, it gives you a list of semantic arguments (Table 5.1, page 76). To remove a verb-placeholder or a mapping, click [Remove] button on the right (Fig. 5.12).

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 5.5: List of syntactic sources

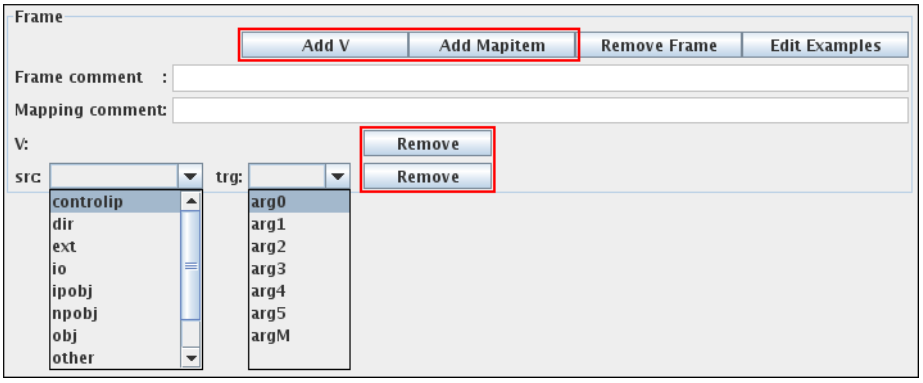

Figure 5.12: Add, edit and remove mappings

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. 5.13).

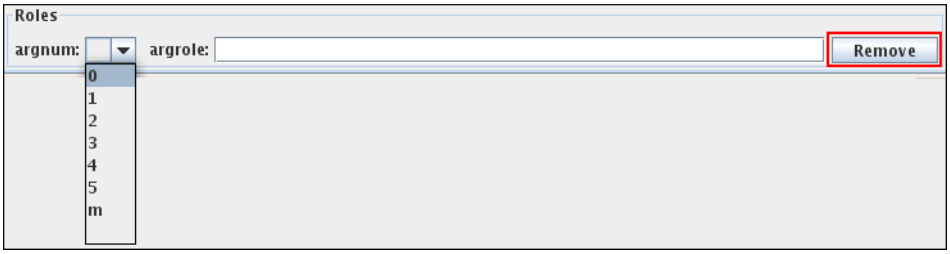

Figure 5.13: Edit and remove an argument

5.4.3 Edit examples

To view examples of the frame, click the `[Edit Examples]` button (Fig. 5.11, page 85), which will prompt a new window: the example frame. Fig. 5.14 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+−`).

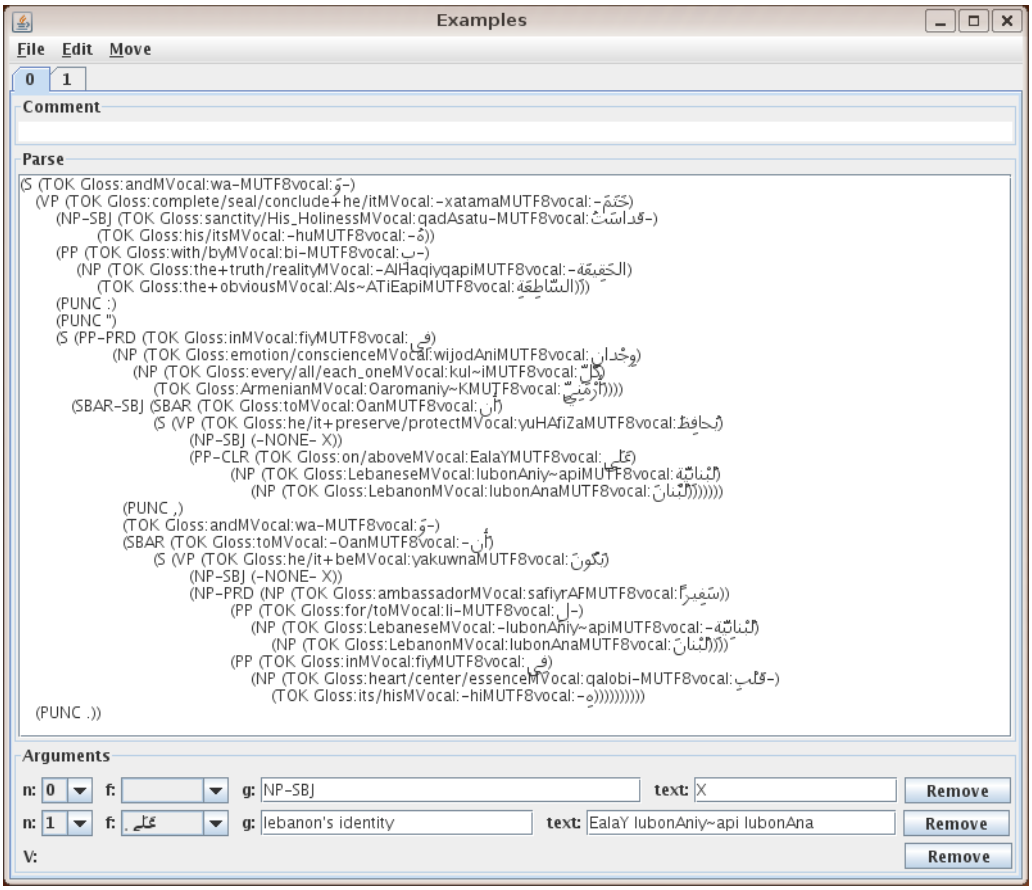

Figure 5.14: Example frame for `f2` in `HAfaZ.xml`

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 5.3.3), 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.

5.4.4 Save a frameset file

Your work will not be saved unless you assign the job to Cornerstone. 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.

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 (COLING-ACL'98)*.

Miriam Butt. 2003. The light verb jungle. In C. Bowern G. Aygen and C. Quinn, editors, *Papers from the GSAS/Dudley House Workshop on Light Verbs.*

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)*.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(1):547–619.

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)*.

Beth Levin and M. Rappaport Hovav. 1995. *Unaccusativity: At the Syntax-Lexical Semantics Interface; Linguistic Inquiry Monograph.* MIT Press.

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.

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.