SEMANTICS EXTRACTION FROM TEXT

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Technology

by

Prashanth K
(Roll No. 08305006)

Under the guidance of
Prof. Pushpak Bhattacharyya

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY–BOMBAY
2010

Abstract

The main focus of this work is to improve the English to Universal Networking Language (UNL) enconversion system built at IIT Bombay by studying grammar, observing various language phenomena and implementing them as rules to generate UNL graphs. We have carried out an error analysis of the system and implemented a few system enhancements and improved the knowledge and rule bases. Our main observation is that the system does not generate good UNL graphs for long sentences. Based on this observation, we have implemented a text simplifier, which breaks a long sentence into smaller sentences. We generate UNL for each of these smaller sentences and propose a technique for merging the generated UNLs.

Acknowledgments

First of all, I would like to thank my guide Prof. Pushpak Bhattacharyya for his invaluable support and guidance. I am also grateful to Avishek Ghosh for giving me a nice introduction to the system modules and the work flow. I also extend my thanks Ashutosh Agarwal and Praveen Lakhotia for their invaluable suggestions and comments. Lastly, I thank my parents for their constant support and encouragement.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the most important and difficult research areas in modern times is to understand human natural language. Though there are a number of systems which deal in specialized areas like ticket booking and financial transactions, domain independent natural language understanding systems are still a matter of research.

Before natural language understanding systems can be created, the meaning of the sentence at a shallow level must be understood. Semantics extraction from text is the process of identifying the semantic relationship between the syntactic constituents of a sentence and its predicate. For example take the following sentence.

The boy read the book.

In this sentence the main predicate is *read*. Given this sentence, the job of a semantics extraction system is to find the relationship of the other parts of the sentence with respect to the predicate *read*. For this sentence, the semantics extraction system will find that the phrase *the boy* is the doer of an action which, in this case, is reading. The doer of the action has performed the action on something which, in this case, is *the book*.

Thus semantics extraction is the process of understanding the meaning from a given text by finding out the various events in which each part of the sentence takes part with respect to the main predicate.

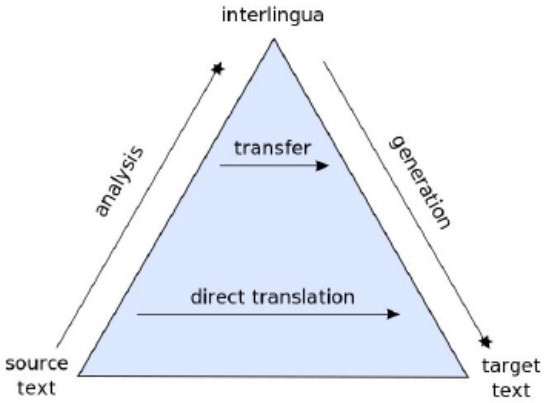


Figure 1.1: Pyramid showing comparative depths of intermediate representation

1.1 Motivation

The world has been witnessing an information revolution for the last decade or so. With the advent of the Internet and the World Wide Web, colossal amounts of information is now available over the Web, and is getting bigger and bigger all the time. Moreover, the world of Internet is getting more and more multilingual. Though English continues to remain the most widely used language over the Web, there is a significant amount of information in other languages as well (notably Chinese, Spanish, Japanese). This has resulted in an increasing need for Machine Translation and Multilingual Search systems.

Machine Translation (MT) systems can be broadly classified into two categories. Statistical MT systems have proved to be highly robust and scalable; on the other hand, knowledge or rule-based systems are successful in providing high quality in translation. Several methodologies exist for Machine Translation. Based on the depth of the processing done on the source content, they can be divided into direct, transfer-based or interlingua-based approaches. Each of these have their own advantages and shortcomings. Figure 1.1 depicts the various paradigms of MT based on depth of analysis and the interlingua.

As mentioned, one of the approaches for performing Machine Translation is Interlingua-based. An interlingua is a language-independent form of representation of the content from the natural language, and can be used as an intermediate form in translating from a source language to a target language (Figure 1.2). The process of getting to the interlingua from the source lan-

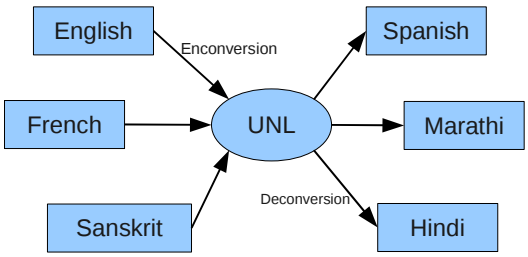


Figure 1.2: UNL as Interlingua

guage text is called the *Enconversion*, while the process of generating the target language text from the interlingua is called *Deconversion*. The choice of interlingua is an important consideration in interlingua based Machine Translation. The interlingua must be able to preserve the semantic information present in the source text and also, there should be no ambiguity in the information it represents. Extensive semantic analysis needs to be done on the source text to be able to capture into the interlingua all the information the source represents. An important standardized Interlingua which is not restricted to any specic domain is the Universal Networking Language (UNL)[UND04]. It is capable of representing semantics of natural language content in an unambiguous manner. It represents the information in a hypergraph structure, in which universal words annotated with zero or more attributes are connected with each other by semantic relations. We describe UNL in greater detail in Chapter 2. Semantics extraction from text also finds application in question answering and text entailment systems. It can also be used in automatic text summarization, text data mining and information extraction systems.

1.2 Problem Statement

The aim of this project is to improve the existing English to UNL conversion system that has been under steady development in Centre for Indian Language Technologies, IIT Bombay for the last few years. This is to be done by studying various aspects of the language, discovering language phenomena and implementing them.

1.3 Organization of this report

In Chapter 2, we present a brief overview of UNL. In Chapters 3 and 4, we describe the current system in detail. Chapters 5, 6 and 7 detail the work we have done. Chapter 5 enlists the problems with the current system and some solutions we have implemented. Chapters 6 and 7 present the architecture of the text simplifier and UNL merger we implemented to solve the long-sentence problem described in Chapter 5.

Chapter 2

Universal Networking Language

The Universal Networking Language (UNL)[UND04] is an ambitious initiative launched in 1996 by The Institute of Advanced Studies of United Nations University and was later inherited by the UNDL Foundation[UND10] in 2001.

UNL is an Interlingua, which represents information in an unambiguous manner, unlike natural languages. It is an artificial language that replicates the functions of natural language and enables computers to process information and knowledge. It works as a Language Infrastructure (LI) and helps in distributing, receiving and understanding natural language. It also acts as a Knowledge Infrastructure which helps computers to process information and make intelligent decisions based on them. This property makes it useful for multilingual machine translation.

UNL represents information in the form of directed semantic graphs with hyper-nodes. Each of the nodes represents an unambiguous concept and the edges represent the relationship that they form among themselves. The nodes can themselves be composed of semantic graphs

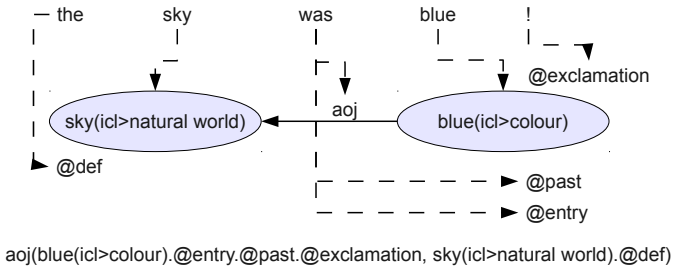


Figure 2.1: Constructing a UNL graphs

representing a concept, and hence the name hyper-nodes.

Figure 2.1 shows the UNL graph for the sentence *The sky was blue!*. The serialized representation of the graph is as under -

```
[unl]
aoj(blue(icl>colour).@entry.@past.@exclamation,sky(icl>natural world).@def)
[/unl]
```

2.1 Elements of UNL

To represent natural language, UNL needs to have all the features corresponding to a natural language. It has to make sure that the knowledge is expressed in an unambiguous manner. Hence, it is composed of words called *Universal Words* which express concepts in an unambiguous manner. These Universal Words (UW) are inter-linked with each other to form a sentence. These links are known as *relations* which, along with the UWs, express the complete meaning of a sentence. The subjectivity of a sentence is expressed through *attributes*.

Next, we will look at each of the components of UNL in detail.

2.1.1 Universal Words

They are not only the syntactic and the semantic unit of the UNL but also the basic element by which the UNL graph of a sentence or a compound concept is constructed. They constitute the vocabulary of UNL and are similar to the concept words in natural language. The only difference between an UW and a word in natural language is that the UW unambiguously represents a single concept but a word in a natural language can have many different concepts in different contexts.

The format of the UW is as follows:

```
< UW >::= < headword > [< constraintlist >]
```

It is made up of a headword which is an English word and may represent many different concepts. The constraint list limits this headword to only one of the concepts giving the UW its unambiguous form.

The universal words can be essentially divided into six different types.

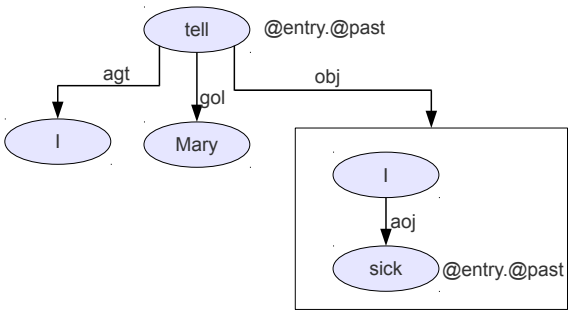


Figure 2.2: A UNL graph containing a compound UW

1. Basic UWs: These are the bare headwords without any constraint list. They are used when the English expression has no ambiguity.
   Examples: `go`, `take`, `house`.

2. Restricted UWs: The English expressions in these cases are followed by a constraint list which helps to bring out a more specific concept of the English expression. Example:

   ```
   state(icl>region), state(icl>abstract thing)
   ```

3. Extra UWs: They help to represent concepts that are not found in English. They use the foreign language word transliterated in English as the head word. Example:

   ```
   ikebana(icl>flower arrangement)
   ```

4. Temporary UWs: Things like address or phone number do not represent any particular concept and hence there is no need to define them. These types of UWs are called temporary UWs
   Example: `1234`, `http://www.iitb.ac.in`.

5. Compound UWs: They are a set of relations grouped together to represent a compound concept. Example: I told Mary *that I was sick*. (Figure 2.2)

6. Null UWs: It is an empty string used to represent ellipsis
   Example: John was killed.

   ```
   agt(kill(agt>thing,obj>thing).@entry.@past, " ")
   obj(kill(agt>thing,obj>thing).@entry.@past, John.@topic)
   ```

2.1.2 Relations

The semantic network is composed of binary relations, each of which is composed or two UWs and a relation that links this UWs semantically.

UNL Relations are chosen based upon two principles. The first principle imposes a necessity condition which states that a relation is necessary if one cannot do without it. The second principle imposes a sufficiency condition on the set of relations. It states that the whole set of UNL meets this constraint when there is no need to add any other relation to this set. Adding of any other relation will produce the condition where there are two relations representing the same concept.

A binary relation is expressed in the following format

```
< relation > (< uw1 >;< uw2 >)
```

This means that the universal word given in *uw*2 hold the relation <relation> with the universal word given in *uw*1.

The UNL relations are hyper semantic, i.e. each node in the graph can be replaced with another network. Such a node is called a scope. The general description format of binary relations is of the following format

```
< relation >:< scope-id > (< node1 >;< node2 >)
```

Here <node1>or <node2>can be a scope node representing a compound concept. In case of a scope node instead of a UW it is represented with its scope-id.

Some examples of UNL Relations are as follows:

1. agt (Agent): It defines a thing that initiates an action
   Example Sentence: John breaks ...

   ```
   agt(break(agt>thing,obj>thing), John(iof>person))
   ```

2. obj (Affected thing): It defines a thing in focus that is directly affected by an event or state Example Sentence: to cure the patient

   ```
   obj(cure(agt>thing,obj>thing),patient(icl>person)
   ```

2.1.3 Attributes

Attributes express the subjectivity of the sentence from the speaker's point of view. While UWs and relations describe the objectivity of the sentence, attributes bring out the way the speaker wants to portray the meaning of the sentence.

It is used to express the following kinds of information

1. Time with respect to the speaker
   Example: `@past` (happened in the past); It was raining yesterday.

2. The speaker's view of aspect
   Example: `@begin` (beginning of a state or event); It began to rain again.

3. Reference to a range of concept
   Example: `@def` (already referred); the book you lost.

4. The speaker's view of emphasis, focus and topic
   Example: `@entry` (entry or main UW of a sentence or a scope); He promised that he would come.

2.2 Compound Universal Words

Compound UWs represent concepts that are a semantic hyper-node graph by themselves. Each of the Compound UW is composed of many UWs and relations among themselves. They are used when a particular part of a sentence forms a relationship with another universal word. Then this whole part is represented as a compound UW.

For example take the sentence, *I told Mary that I was sick.*

Here, *I was sick* forms a relation **obj** with *tell*. *I was sick* hence is represented as a compound UW (Figure 2.2). The serialized UNL for this sentence is as under. The scope id *:01* represents the compound UW *I was sick*.

```
[unl]
agt(tell:2.@entry.@past,I:0)
aoj:01(sick:11.@entry.@past, I:7)
```

```
gol(tell:2.@entry.@past,Mary:4)
obj(tell:2.@entry.@past,:01)
[/unl]
```

For an exhaustive treatment of UNL, please refer to the UNL manual[UND04].

2.3 Summary

In this chapter, we gave a brief overview of UNL and how it can be used to represent the meaning of a sentence unambiguously. In the next chapter, we describe the working of our English-to-UNL Enconverter.

Chapter 3

Enconverter system

Automatic Generation of UNL based on XLE Parser was developed by Sandeep Limaye[Lim07] and improved upon by Avishek Ghosh[Gho08]. This chapter gives a brief description of the system.

Figure 3.1 shows the work-flow of our Enconverter. We can divide the work-flow into three stages.

1. Word Entry Collection Generation
2. Word Entry Collection Enhancement
3. UNL Generation

These stages are described in detail in the following sections. For all the stages, we explain the work flow taking the sentence *"Rafael Nadal ate the juicy oranges in the kitchen hastily"* as example.

3.1 Word Entry Collection Generation

The word entry collection (WEC) is a collection of words of the sentence along with their syntactic and semantic properties and relations with other words of the sentence. In this stage, we collect the content words of the sentence and generate properties for each of them using the F-structure[Dal01] from XLE parser[CDK$^+$08]. This helps in UNL relation and attribute generation. We use a WSD system for getting the sense of the words, a Named Entity Recoginition

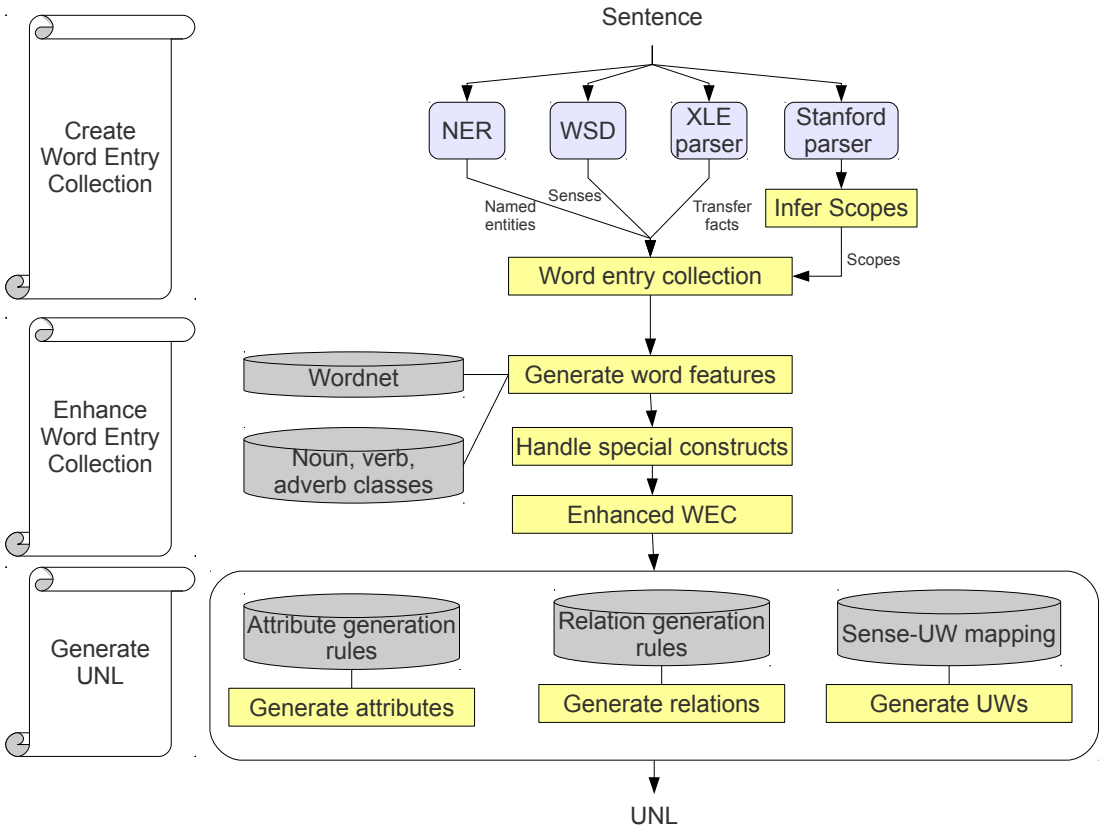


Figure 3.1: Work-flow of Enconverter

System to recoginize people, places and organizations, and a constituency parser to infer the compound UWs (or UNL scopes).

In the following subsections, we explain the contribution of each of the systems towards generation of the WEC.

3.1.1 XLE parser

The XLE praser outputs the F-structure as a set of triplets, each is called a transfer fact. The transfer facts for the example sentence are[1] -

```
ADJUNCT,eat:5,hastily:13
ADJUNCT,eat:5,in:10
ADJUNCT,orange:9,juicy:8
ADV-TYPE,hastily:13,vpadv
ATYPE,juicy:8,attributive
CASE,Nadal:1,nom
CASE,kitchen:12,obl
CASE,orange:9,obl
CLAUSE-TYPE,eat:5,decl
COMMON,kitchen:12,count
COMMON,orange:9,count
DEGREE,hastily:13,positive
DEGREE,juicy:8,positive
DET,kitchen:12,the:11
DET,orange:9,the:7
DET-TYPE,the:7,def
DET-TYPE,the:11,def
HUMAN,Nadal:1,+
HUMAN,Rafael:0,+
MOOD,eat:5,indicative
NSYN,Nadal:1,proper
```

[1] XLE parser terminology explained in Appendix A

```
NSYN,Rafael:0,proper
NSYN,kitchen:12,common
NSYN,orange:9,common
NUM,Nadal:1,sg
NUM,Rafael:0,sg
NUM,kitchen:12,sg
NUM,orange:9,pl
OBJ,eat:5,orange:9
OBJ,in:10,kitchen:12
PASSIVE,eat:5,-
PERF,eat:5,-
PERS,Nadal:1,3
PERS,Rafael:0,3
PERS,kitchen:12,3
PERS,orange:9,3
PROG,eat:5,-
PROPER-TYPE,Nadal:1,name
PROPER-TYPE,Rafael:0,name
PSEM,in:10,set_valued:28
PTYPE,in:10,sem
SUBJ,eat:5,Nadal:1
TENSE,eat:5,past
VTYPE,eat:5,main
LEX-SOURCE,Nadal:1,guesser
LEX-SOURCE,Rafael:0,morphology
LEX-SOURCE,hastily:13,morphology
LEX-SOURCE,juicy:8,morphology
LEX-SOURCE,kitchen:12,countnoun-lex
LEX-SOURCE,orange:9,morphology
SUBCAT-FRAME,eat:5,V-SUBJ-OBJ
```

Here, `ADJUNCT, eat:5, hastily:13` says that *hastily* is an adjuct of *eat*. `ATYPE, juicy:8, attributive` says that *juicy* is an attributive adjective. The other triplets can be interpreted similarly. For every word with a unique ID in the transfer facts list, we create an entry in the word entry collection and add the corresponding features to the word entry. For example, for `ATYPE, juicy:8, attributive`, we create the following word entry -

Word: juicy:8 **Features:** ATYPE:adjective

If we encounter another transfer fact for *juicy:8*, we add the new feature to the feature list. Thus, the WEC is initialized. The WEC for the above set of transfer facts is

```
the:11,     (DET-TYPE:def )
rafael:0,   (HUMAN:+ LEX-SOURCE:morphology NSYN:proper PERS:3 PROPER-TYPE:name
             NUM:sg )
the:7,      (DET-TYPE:def )
juicy:8,    (LEX-SOURCE:morphology ATYPE:attributive DEGREE:positive )
in:10,      (PTYPE:sem PSEM:set-valued:28 OBJ:kitchen:12 )
kitchen:12,(LEX-SOURCE:countnoun-lex DET:the:11 COMMON:count NSYN:common PERS:3
             NUM:sg CASE:obl )
orange:9,   (LEX-SOURCE:morphology DET:the:7 COMMON:count ADJUNCT:juicy:8
             NSYN:common PERS:3 NUM:pl CASE:obl )
eat:5,      (VTYPE:main PROG:-_ ADJUNCT:hastily:13 in:10 PASSIVE:- PERF:-
             SUBCAT-FRAME:v-subj-obj OBJ:orange:9 CLAUSE-TYPE:decl
             MOOD:indicative TENSE:past SUBJ:nadal:1 )
nadal:1,    (HUMAN:+ _LEX-SOURCE:guesser NSYN:proper PERS:3 NAME-MOD:rafael:0
             PROPER-TYPE:name NUM:sg CASE:nom )
hastily:13,(LEX-SOURCE:morphology ADV-TYPE:vpadv DEGREE:positive )
```

3.1.2 Named Entity Recogintion

The NER system identifies people, places and organizations in a sentence. The corresponding features are added to the WEC. If a named entity spans multiple words in the WEC, it is reduced to a single entry with new unique ID, the feature lists are combined and every reference to any of these words in the word entry collection is changed to a reference to the new single entry. For example, *Rafael Nadal* is recognized as a name. Word entries `rafael:0` and `nadal:1` are

replaced by the following word entry.

```
rafael_nadal:1001, (HUMAN:+ _LEX-SOURCE:morphology NSYN:proper PERS:3
                    NUM:sg PROPER-TYPE:name )
```

3.1.3 Word Sense Disambiguation

Every word in WEC which is not a named entity is disambiguated and its corresponding WordNet[Fel98] synset ID is added to the feature list of that word.

3.1.4 Stanford Parser

We use stanford parser to generate scopes (or compound UWs). We discuss scope generation in Section 3.4.

3.2 Word Entry Collection Enhancement

In this stage, we enhance the WEC by adding more features to it, and by handling special language phenomena.

3.2.1 Adding additional features to word entries

To generate additional word features, we use the word senses and the named entities given by the WSD and NER systems and also a few word class databases extracted from WordNet and other resources.

Features for common nouns

We have classified all common nouns into 30 different classes (Sample nouns and their classes are shown in Table 3.1). A word may belong to multiple classes. For example, the noun *duck* may belong to any one of the following classes - *noun.animal, noun.artifact, noun.food, noun.quantity*. Using the sense ID of the word and WordNet, we determine the correct class of the word in the context. Then, we add the features associated with the class to the word entry from word class tables built from WordNet.

For our example sentence, the noun class of *orange* is *noun.food*. So, we add the feature *CONCRETE* to the feature list. Table 3.2 shows noun classes and associated features.

Table 3.1: Nouns and classes

noun	**nounclass**
Zyrian	noun.communication
zymurgy	noun.cognition
zymosis	noun.process
zymolysis	noun.process
zymogen	noun.substance
zymase	noun.substance

Table 3.2: Noun class - Features mapping

nounclass	**nounclassID**	**features**
noun.act	n004	EVENT,ABSTRACT
noun.animal	n005	ANIMATE,CONCRETE
noun.person	n006	PERSON,ANIMATE
noun.attribute	n007	ABSTRACT
noun.food	n008	CONCRETE

Features for named entities

For named entities, we reuse the common noun classes mentioned earlier. We assign the class *noun.location* for places, *noun.person* for people and *noun.location, noun.group* for organizations. We then add the features the same way as we did for common nouns. For `rafael nadal` in our example sentence, we assign the class *noun.person* and hence generate the features `PERSON`, `ANIMATE`.

Features for verbs

We have classified verbs into 150 classes. A few example classes are shown in Table 3.3. We have mapped every verb in WordNet to a class (Sample entries are shown in Table 3.4) Here, we add the verb class as a feature for the word entry.

Table 3.3: Content of table verb_class_mapping

verbclass	**name**
v001	Verbs of Instrument of Communication
v002	Advise Verbs
v003	Talk Verbs
v004	Complain Verbs
v005	Tell verb

Table 3.4: Verbs and classes

verb	**verbclass**
abase	v135
abash	v127
abate	v135
abate	vErg
abbreviate	v135
abdicate	v041

Features for adverbs

We have classified adverbs into adverbs of place, time and manner (Sample entries in Table 3.5). We perform a table look-up to determine the class of an adverb. We use the class itself as the feature here.

Table 3.5: Adverbs and classes

adverb	adverbclass
a_priori	TIME
aboard	PLACE
above	PLACE
abroad	PLACE
abruptly	MANNER
absolutely	MANNER

3.2.2 Handling special language phenomena

Since this involves a deep study of XLE parser and language, and also interrupts the flow of our discussion, we discuss this in Chapter 4.

3.3 UNL Generation

There are 3 steps to generate the UNL of a sentence.

- Attribute generation
- UW generation
- Relation generation

We discuss each of these in detail in the following subsections.

3.3.1 Attribute generation

We use the features in the WEC feature list to generate attributes for words. We use a mapping from WEC features to UNL attributes as shown in Table 3.6. A few other attributes are generated during the Relation Generation step. Example: `@pl` for `orange:9`, is generated using row 6 of the table. Note that `orange:9` contains `NUM:pl` in its feature list.

Table 3.6: Transfer fact - Attribute mapping

transfer_fact	value_attribute
ADJUNCT-TYPE	neg:@not
DET-TYPE	def:@def, indef:@indef
EMPHASIS	+:@emphasis
_INV	+:@not
MOOD	imperative:@imperative
NUM	pl:@pl
PERF	+_:@complete
PROG	+_:@progress
TENSE	fut:@future, past:@past, pres:@present

3.3.2 UW Generation

We use the sense IDs generated from WSD for generating UWs. We use a table mapping WordNet sense to UNL UW. A few sample entries are shown in Table 3.7 If the word does not have a sense ID (for proper nouns etc), we use the word itself as UW.

Table 3.7: WordNet sense - UW mapping

uw	sense_id
aah(icl>exclaim>do, equ>ooh, agt>thing)	aah%2:32:00::
abacinate(icl>blind>do, agt>thing, obj>thing)	abacinate%2:39:00::
abandon(icl>discard>do, agt>thing, obj>thing)	abandon%2:40:00::
abandon(icl>do, agt>thing, obj>thing)	abandon%2:40:01::
abandon(icl>leave>do, equ>vacate, agt>thing, obj>thing)	abandon%2:38:00::
abandon(icl>leave>do, agt>thing, obj>thing)	abandon%2:31:00::

3.3.3 Relation Generation

We generate relations based on

1. Subcategorization frames

2. Adjuncts

3. Prepositional adjuncts

A subcategorization frame gives us the arguments that a verb takes. Any part of the sentence that does not fit into the subcat frame of the verb is called the *Adjunct*. For example, `eat` has a subcat frame `V-SUBJ-OBJ`, meaning it takes a subject and an object as arguments. The part *in the kitchen* of our sentence does not fit into the subcat frame, and hence is an adjunct.

The method we adopt for generating relations is described below -

Based on subcat frames

The attribute list for `eat` is

```
VTYPE:main PROG:- ADJUNCT:hastily:13 in:10
PASSIVE:- PERF:- SUBCAT-FRAME:v-subj-obj
OBJ:orange:9 CLAUSE-TYPE:decl MOOD:indicative
TENSE:past SUBJ:rafael nadal:1001
```

From this, we select the feature SUBCAT-FRAME and read its value (`v-subj-obj`). Now, we look up a table mapping subcat frames to "related transfer facts" (Table 3.8, only a few entries are shown) to determine which UW is related to `eat`. We get `SUBJ` and `OBJ` as related transfer facts for subcat frame `V-SUBJ-OBJ`. By looking at the values of features `SUBJ` and `OBJ` for `eat`, we can infer that `eat` is related to `orange` and `rafael nadal`.

Table 3.8: Content of table subcat_transferfact_map

subcat_frame	**transfer_fact**
V-SUBJ-OBJ	SUBJ
V-SUBJ-OBJ	OBJ
V-SUBJ-OBJ	OBL-AG
V-SUBJ-OBJ-COMPEXopt_extra	SUBJ
V-SUBJ-OBJ-COMPEXopt_extra	OBJ
V-SUBJ-OBJ-COMPEXopt_extra	COMP

To generate the relations, we use the related transfer fact and word features. We have created a table which maps transfer facts and other word features to relations (Sample entries in Table 3.9). For `orange`, we retrieve the relation from the table which is associated the following -

tf_1 is `'OBJ'` or `Null`

w1_prop (i.e. features of `eat`) contains `'VTYPE=main'` and `'PASSIVE=-'`

w2_prop (i.e. features of `orange`) is `'CONCRETE'` or `Null`.

From the table, we get the relation `obj(eat,orange)`. Similarly, we get the relation `agt(eat,rafael nadal)`. We also generate attribute `@entry` for `eat` here.

Table 3.9: Features - Relation mapping

subcat_frame	tf_1	w1_prop	w2_prop	relation	uw1	uw2	attrib
V-SUBJ-OBJ-OBL	OBJ	VTYPE=main PASSIVE=-	ANIMATE	gol	1	2	Null
Null	SUBJ	VTYPE=main PASSIVE=- DOTYPE=do	Null	agt	1	2	Null
V-SUBJ-OBJexpl-XCOMPPREDap	XCOMP-PRED	VTYPE=main PASSIVE=-	Null	aoj	2	3	@entry
V-SUBJ-OBLloc	OBL	VTYPE=main PASSIVE=-	Null	plt	1	2	Null
V-SUBJ-OBJ-OBL	OBJ	VTYPE=main PASSIVE=-	ANIMATE	gol	1	2	Null
V-SUBJ-OBJ-OBL	OBJ	VTYPE=main PASSIVE=-	Null	obj	1	2	Null

The numbers under columns **uw1** and **uw2** give the direction of the relation. $uw1 = 1$ and $uw2 = 2$ gave us `agt(eat,rafael nadal)` whereas $uw1 = 2$ and $uw2 = 1$ would have given us `agt(rafael nadal,eat)`.

Based on adjuncts

The list of all transfer facts which indicate the presence of adjuncts is retrieved from a table (sample entries in Table 3.10).

Table 3.10: List of adjunct transfer facts

w1_transfer_fact	w2_transfer_fact_prop
DET	DET-TYPE

Table 3.10: List of adjunct transfer facts (continued)

w1_transfer_fact	**w2_transfer_fact_prop**
ADJUNCT	_ADJMOD
ADJUNCT	ADJUNCT-TYPE
ADJUNCT	ADV-TYPE
ADJUNCT	ATYPE
ADJUNCT	DEVERBAL

Every word entry in the WEC is inspected for the presence of the transfer facts in column **w1_transfer_fact**. If any of these transfer facts is present in the feature list of a word entry, we retrieve the value of this feature (which will be reference to a word in WEC). For this word, we retrieve the value of the feature specified in **w2_transfer_fact_prop**. Using this we query yet another table (Sample entries in Table 3.11) to get the associated relation and attributes, if any.

In our case, let us say that we are currently processing the rule in row 4 of Table 3.10. The word `eat` has the trasferfact `ADJUNCT` in its feature list. So, we get the value of the feature `ADJUNCT` which is `hastily:13`. Then we query Table 3.11 for a relation using the features of `eat` and `hastily` i.e we find a relation which is associated with the following

transfer_fact_1 is `'ADJUNCT'`

w2_prop contains `'ADV-TYPE=vpadv'` or `'ADV-TYPE=*'`

w2_features is `'MANNER'`.

From this, we get the relation `man(eat,hastily)`. Note the use of `'ADV-TYPE'`. The column **w2_transfer_fact_prop** of Table 3.10 instructed us to use this feature of `hastily:13` to find the relation.

Table 3.11: Adjunct relation rules

transfer_fact_1	**w2_prop**	**w2_features**	**relation**	**uw1**	**uw2**	**attrib**
ADJUNCT	ADJUNCT-TYPE=neg	TIME	tim	1	2	Null
ADJUNCT	ADV-TYPE=focus ADV-TYPE=sadv ADV-TYPE=vpadv	TIME	tim	1	2	Null

Table 3.11: Adjunct relation rules (continued)

transfer_fact_1	w2_prop	w2_features	relation	uw1	uw2	attrib
ADJUNCT	ADV-TYPE=focus ADV-TYPE=sadv ADV-TYPE=vpadv	MANNER	man	1	2	Null
DET	DET-TYPE=indef	Null	Null	Null	Null	@indef
DET	DET-TYPE=def	Null	Null	Null	Null	@def
ADJUNCT	_ADJMOD=noun	Null	mod	1	2	Null

Based on prepositional adjuncts

If the adjunct contains a preposition (For example, "*in the kitchen*"). We determine the relation based on the class of the main verb (as obtained during feature generation, see section 3.2.1), the part of speech tag of the verb, the object of the preposition, and the preposition itself. In our example, we generate `plc(eat,kitchen)` by looking at the following -

syncat1 (word 1 is `eat`) = `'V'`

semcat1 = `'vUnErgDo'`

prep_lexitem = `'in'`

syncat3 (word 3 is `kitchen`)= `'N'`

semcat3 = `'CONCRETE'` or `'ARTIFACT'`.

We match these feature values against Table 3.12 (which shows a few of the rules we use for generating relations for prepositional adjuncts) to get the required relation.

Table 3.12: Relation rules for Prepositional adjuncts

syncat1	semcat1	prep_lexitem	syncat3	semcat3	relation	uw1	uw2
V	v001	about	N	-	obj	1	3
V	v001	to	N	-	gol	1	3
V	v002	about	N	-	obj	1	3
V	v002	against	N	-	obj	1	3
V	v002	on	N	-	obj	1	3

Table 3.12: Relation rules for Prepositional adjuncts (continued)

syncat1	semcat1	prep_lexitem	syncat3	semcat3	relation	uw1	uw2
V	vUnErgDo	in	N	CONCRETE	plc	1	3

3.3.4 Putting things together

The attributes, scopes, relations, UWs discovered so far are finally put together and the UNL is generated.

```
[unl]
agt(eat(icl>eat>do):11.@entry.@past,rafael nadal:10.@entry)
aoj(juicy(icl>adj, ant>juiceless):13,orange:12.@pl.@def)
man(eat(icl>eat>do):11.@entry.@past,hastily(icl>how,equ>hurriedly,com>hasty):14)
obj(eat(icl>eat>do):11.@entry.@past,orange:12.@pl.@def)
plc(eat(icl>eat>do):11.@entry.@past,kitchen(icl>room>thing):15.@def)
[/unl]
```

3.4 Scope (or compound UW) Identification

We use the Stanford Parser for generating scopes (or compound words). Stanford parser uses the Penn Treebank notation in its parse trees. The Penn Treebank[MMS93] has the following five types of nodes that indicate clausal structures in the sentence.

S A simple declarative clause, i.e. one that is not introduced by a (possibly empty) subordinating conjunction or a wh-word and that does not exhibit subject-verb inversion

SBAR A clause introduced by a (possibly empty) subordinating conjunction

SBARQ A direct question introduced by a wh-word or a wh-phrase

SINV An inverted declarative sentence, i.e. one in which subject follows the tensed verb or modal

SQ An inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ

For the purpose of scope generation, the above nodes can be clubbed into two types. One in which the clause in not introduced by a subordinating conjunction and the other in which the clause is introduced by either a subordinating conjunction or a wh-word. Thus S, SINV and SQ will fall under the first category while SBAR and SBARQ comprise the second category. From here on, the algorithm is presented with reference to only S and SBAR nodes. But in reality, they refer to the whole class instead of just S or SBAR.

3.4.1 Using SBAR nodes

Whenever the SBAR node occurs, the node along with all its children is made into a scope. For instance, take the following sentence -

I imagine that they are having a good time.[2]

In the above sentence, the object of the verb *'imagine'* is the clause *'that they are having a good time'* and hence it forms a scope. The parse structure *'that they are having a good time'* comes under an SBAR node and hence is rightly identified as a scope.

The other thing that needs to be done during scope generation is to identify the entry word and the main verb for the scope. In the above case the main verb and the entry word are the same - *'having'*. When an SBAR node forms a scope, the verb within the clause is selected to be the main verb.

Now there could be the case that NP nodes occur as siblings to the SBAR node instead of the SBAR node occurring as the only child to the verb as in the previous case. Take the following sentence as example.

John who lives in Delhi plays football.

As can be seen in the parse tree, *'John'*, an NP, is the sibling of the SBAR which is *'who lives in Delhi'*. In this case, the NP along with the SBAR forms the scope. Thus in the above sentence *'John who lives in Delhi'* forms the scope. But here, instead of the main verb *'lives'*, the noun *'John'* forms the entry word.

But not all types of SBAR include their NP sibling. In the above case, the subordinating conjunction for the SBAR was a wh-word and hence it got included. But if the subordinating conjunction is a preposition, the NP sibling does not feature in the scope. Consider the following

[2]Parse trees of all sentences used in this section are in Section 3.4.4

sentence.

You cannot fool him because he comes here daily.

Here though the NP *'him'* is a sibling of the SBAR *'because he comes here daily'*, it does not get to take part in the scope since the subordinating conjunction for the SBAR is a preposition. The UNL expression below makes the reason clear for not including the NP.

```
agt(fool.@entry.@ability.@not,you)
obj(fool.@entry.@ability.@not,he)
rsn(fool.@entry.@ability.@not,:01)
agt:01(come.@entry, he)
man:01(come.@entry, daily)
plc:01(come.@entry, here)
```

Thus, when including the NP sibling, the *type* of the subordinating conjunction takes the final decision. In some cases, the subordinating conjunction itself has to be considered before the final decision can be taken. Consider the following two sentences.

John came to the city where flowers are sold.

John came to the city when the sun was setting.

In both the sentences, the clause is introduced by a wh-word. In the first sentence, the scope gets formed by the SBAR and its NP sibling, but in the second case, the scope includes only the SBAR node. This is because, in the first sentence the whole clause forms a modifier for *'the city'*, but in the second case the clause has a temporal relation with the main verb *'came'* and does not form any relation with the NP *'city'*. Thus, in the above case the exact word needs to be considered when considering whether to include the NP sibling of the SBAR node.

Apart from NP siblings, no other siblings will get included with the SBAR node when they occur without the presence of another NP sibling. The following sentence provides an example for the case of adjective siblings.

Bolivia is beautiful and striking where there are currently three national parks and eight protected areas.

In the above sentence, the SBAR *'where there are currently three national parks and eight protected areas'* forms the scope. The adjective *'beautiful and striking'* does not get included in the scope.

3.4.2 Using S-nodes

There are two kinds of S clauses. One that is introduced by a *to*, i.e. an infinitive *to*, and the other that is not. In both these cases, if the parent main verb of the clause is a 'be' verb, then all the siblings and its parent till the verb are included with that of the S clause while forming the scope. Take the case of the following sentences.

Built of teakwood and bamboo poles, they are fascinating to see.

India is an excellent place to visit.

In the first sentence, the subject place forms an S clause having a sibling NP. *'Built of teakwood and bamboo poles'* forms the S clause having *'they'* as the sibling NP. Since the parent verb of the S clause is a *be* verb in the form of *'are'*, the NP is considered along with the S node when forming the scope.

In both the first and the second sentences, the object place is formed by an S node which is introduced by an infinitive *to*. And since in both cases, the parent verb of the clause is a *be* verb, all its siblings get included in the scope. If no NP sibling is present, the entry word for the scope in the above cases will be the head of the immediate sibling that gets included with the S clause. If an NP sibling is present, then the head of the NP sibling will form the entry word for the scope.

Now let us consider the case of the normal S clause without any infinitival *to*. When the S clause occurs with a non-*be* verb as its parent verb, no other sibling is considered to form the scope if it is in the object position. But the S clause is in the subject position, then the NP gets included with the S clause to form the scope. Consider the following sentences.

Blinded by the dust storm, they fell into disorder.

The emperor exhausted himself running here.

In the first sentence, the S node is composed of the clause *'blinded by the storm'*. Since it occurs in the subject position, the sibling NP *'they'* is considered along with it to form the scope. As before, when the sibling NP is considered, the head of the sibling NP forms the entry word. Thus, in the above example, the word *'they'* will form the entry word.

In the second sentence, *'running here'* forms the S node and *'himself'* forms its NP sibling. But since the S node occurs in the object position, the NP is not included in the scope.

The entry word for the S node depends on its parent. If the parent of the S node is a PP, then

the entry node for the clause would be the main verb of the clause. Otherwise, it is the head of the noun phrase. For instance, take the following two sentences.

The woman I spoke to said otherwise.

I heard of his having gained a prize.

In the first sentence, the `agt` relation would be formed between the verb *'said'* and *'woman'*. Hence, in the first case, the entry word for the scope would be *'woman'*, which agrees with our rule since the S node does not have a PP as its parent. In the second sentence the relation `obj` would be formed between the verb *'heard'* and *'gained'*. Hence, the entry word would be the verb of the scope, which is *'gained'*. This once again conforms to our rule since the S node has a PP as its parent.

But again, if the S node has a parent NP, then the whole NP structure is made into a scope. Take the example of the following sentence.

Visitors may have the privilege of observing the solar eclipse.

In the parse structure, it can be seen that the S clause *'observing the solar eclipse'* has its parent as an NP which includes *'the privilege of observing the solar eclipse'*. In this case, the whole NP is made into a scope. The entry word is the head of the top-most NP which, in this case, would be the word *'privilege'*.

Now let us look at the cases where the S node is introduced by an infinitival *to*. There are only two cases to look at here. We have already considered the case when the main verb of the parent scope is a *be* verb. When the main verb of the parent scope is a non-be verb we do not include any of the siblings of the S node when creating the scope. For example take a look at the following sentence.

John asked Jack to sweep the floor.

In the above sentence apart from the `agt` relation, two other relations are formed with *'asked'*. One is the *'ben'* relation that gets formed with *'Jack'* and the other is the `obj` relation that gets formed with the clause *'sweep the floor'*. Thus, in the above sentence, only the clause *'sweep the floor'* forms the scope and its siblings are not considered at all. The entry word for the scope is the main verb which, in this case would be *'sweep'*.

But if the parent of the S clause is an NP, then in this case too the whole NP is considered as a clause. For example take the following sentence.

The monument took 10 artists to complete.

In the above sentence, the S node *'to complete'* is contained within the NP *'10 artists to complete'*. Thus, in this case the whole NP forms the scope and the entry word is the head word of the NP which in this case would be *'artists'*.

3.4.3 Using conjuncts

When generating scopes for conjunctions, three cases needs to be considered. The simplest of these cases is when the siblings of the conjunction are any non clausal nodes or any non VP nodes. In these cases, the whole phrase is considered a scope and any one of the siblings becomes the entry word for the scope. Take the case of the following sentence.
John and Jack play football.

Here the conjunction *'and'* has NP nodes as its sibling. Hence, the whole phrase 'John and Jack' is considered to be a scope, and either *'John'* or *'Jack'* is made the entry node.

The next case is when clausal nodes form siblings of the conjunction. This case is a bit more complex than the previous case. Here, each of the siblings are already scopes. Now if the conjunction occurs in the default scope, nothing else needs to be done. For instance, consider the following sentence.
I will go home and you will go to work.

In the above sentence, *'I will go home'* and *'you will go to work'* form clauses under S node, and since the conjunction 'and' already occurs in the position of the default scope, nothing else needs to be done. But if the conjunction does not occur in the default scope, then the conjunction along with its sibling will form another scope. This can be illustrated with the help of the following sentence.
He said Jack will go home and Jill will go to work.

In the above sentence, *'Jack will go home'* and *'Jill will go to work'* form clauses under the S node, but unlike the previous sentence, the conjunction *'and'* does not occur in the default scope. Hence the conjunction along with its children forms another scope. Any one of the clauses in both the above cases becomes the entry word. The last case is the case when the siblings of the conjunctions are VPs. For example take the case of the following sentence.
John will water the garden and mow the lawn.

In the above sentence, the conjunction *'and'* has two VP siblings. In this case, each of the

VPs forms a scope. Thus, in the above sentence, *'water the garden'* will form a scope and *'mow the lawn'* will also form a scope. Since in this case, the conjunction can never be in the default scope, the conjunction along with its siblings will form another scope.

3.4.4 Parse trees of sentences used in this section

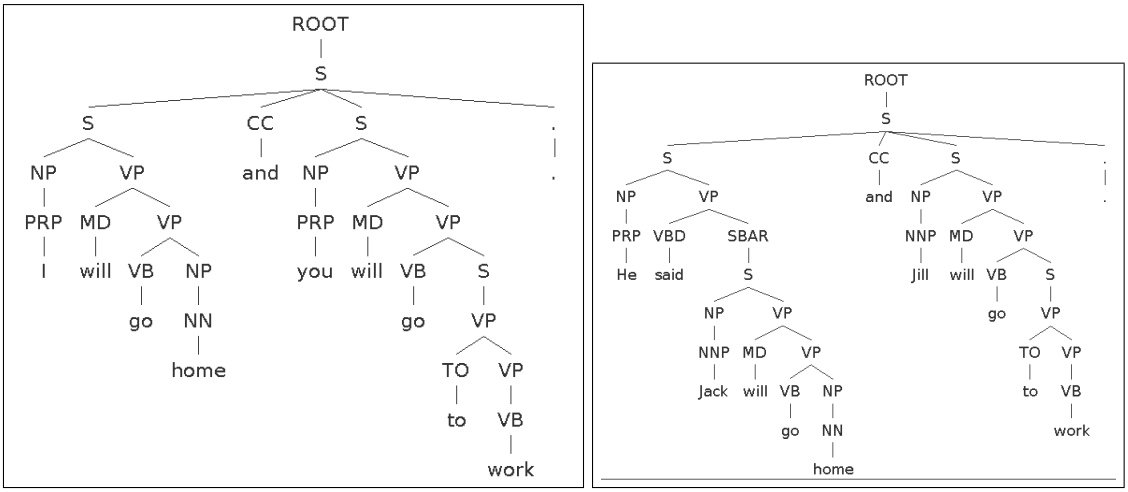
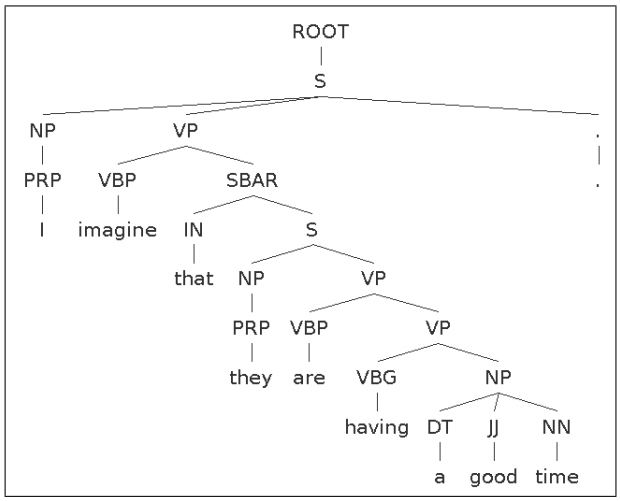
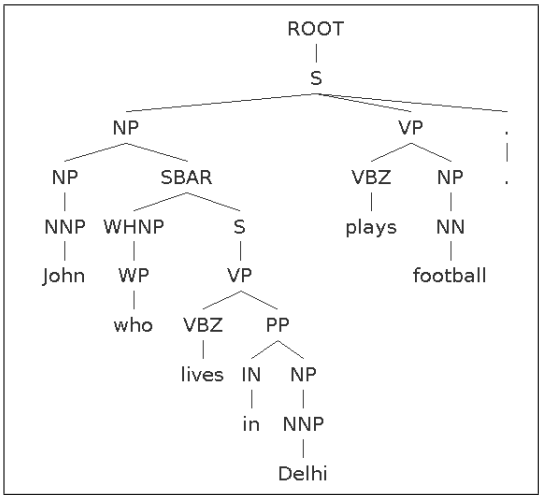


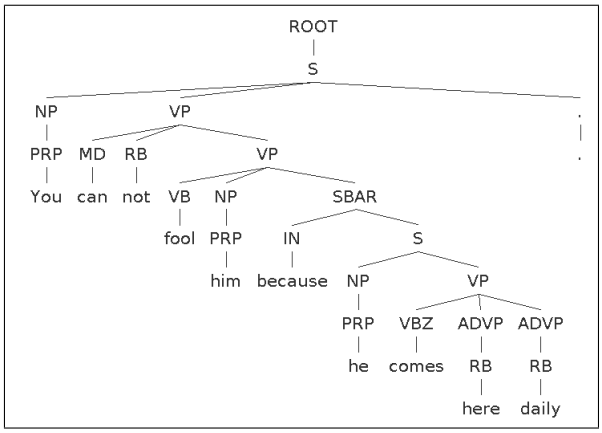
Figure 3.2: Stanford parses of sentences used

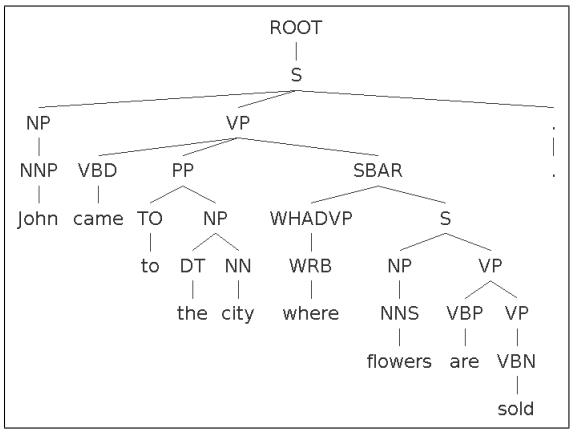
3.5 Summary

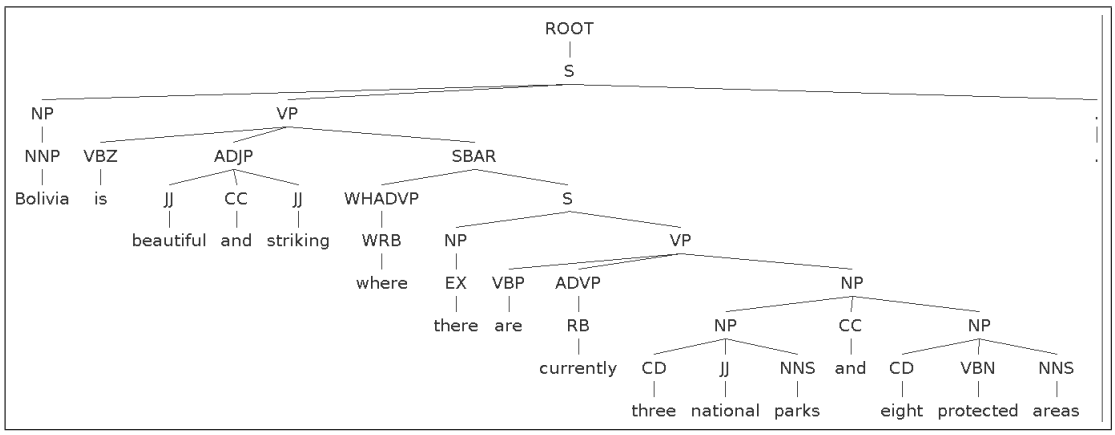
In this chapter, we saw how UNL can be generated in 3 steps - WEC Generation, WEC Enhancement and UNL Generation. We also dedicated an entire section to scope identification. In the next chapter, we will look at the part of WEC Enhancement we left out in this chapter - Handling special language phenomena.



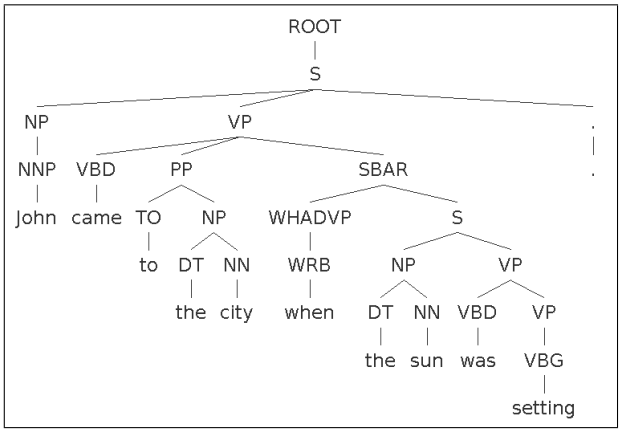


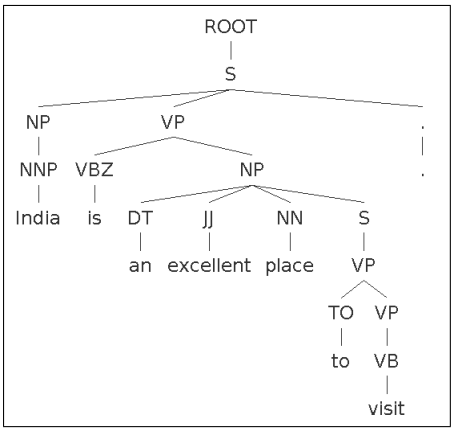


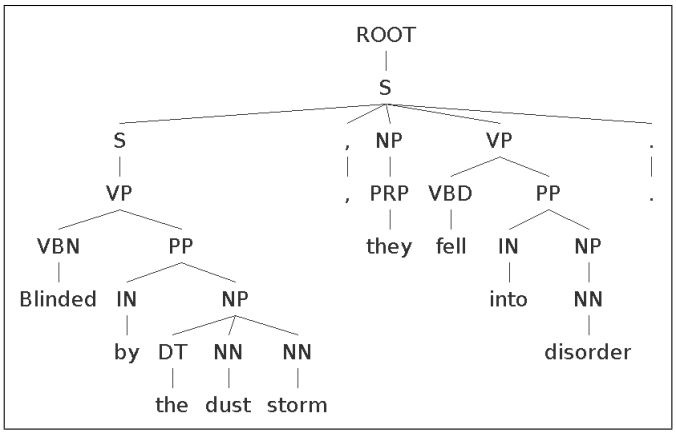


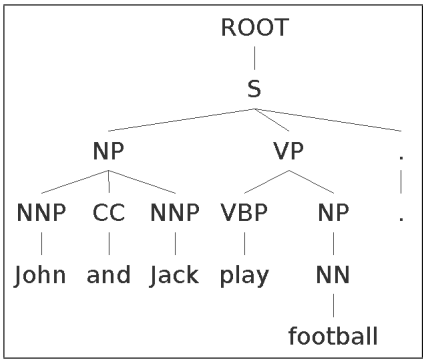


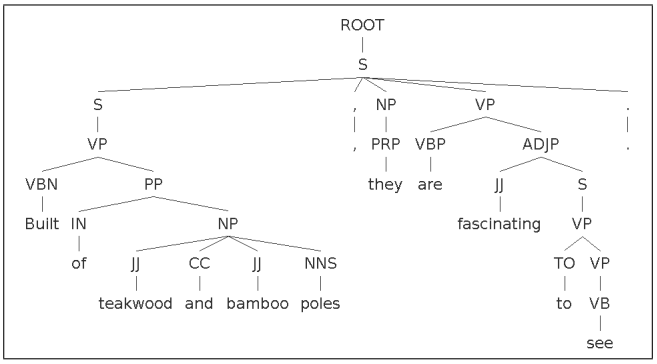
Stanford parses of sentences used (contd.)

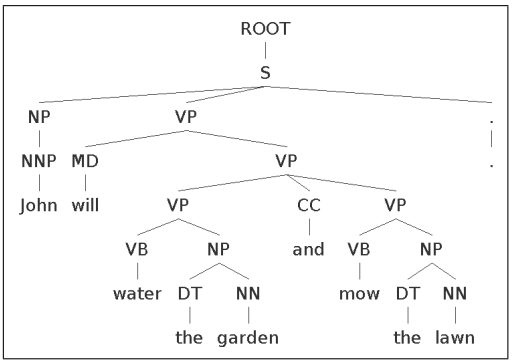




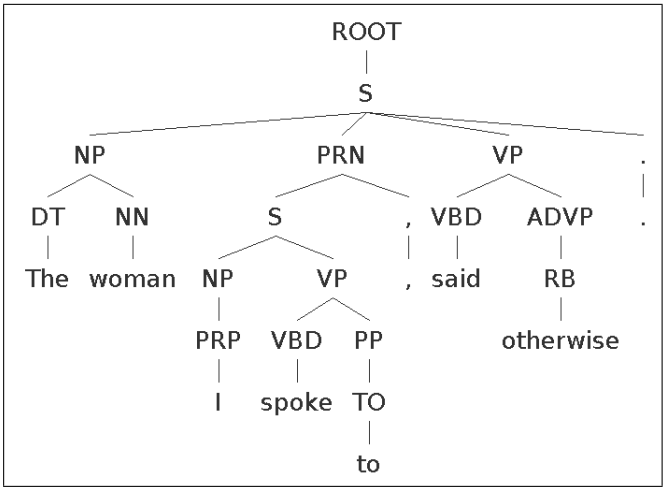


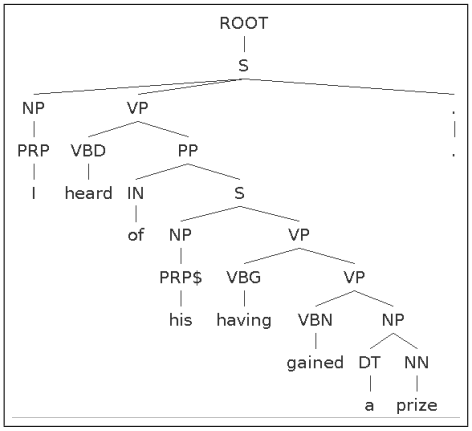


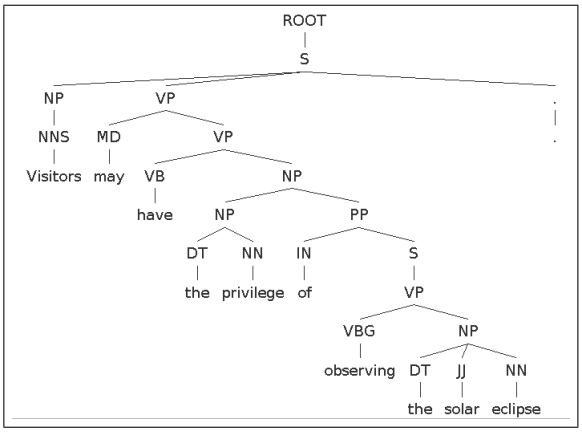


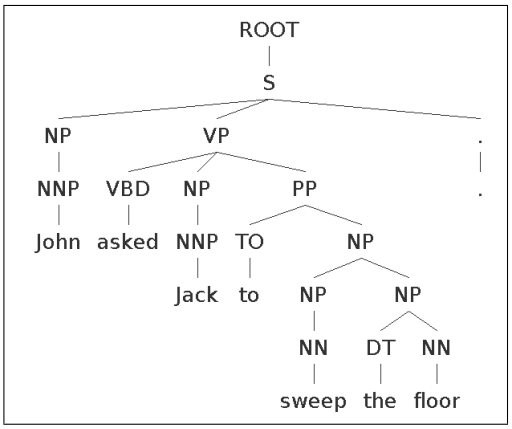


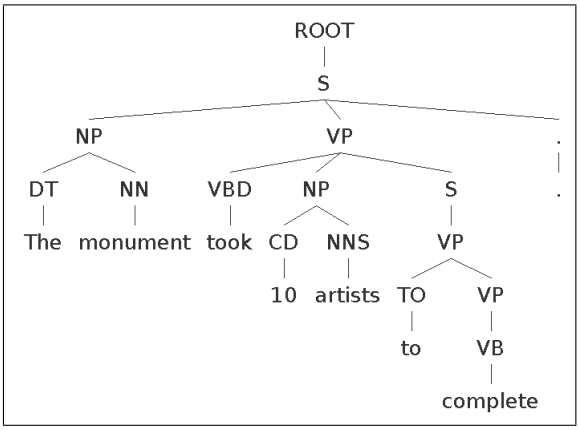
Stanford parses of sentences used (contd.)

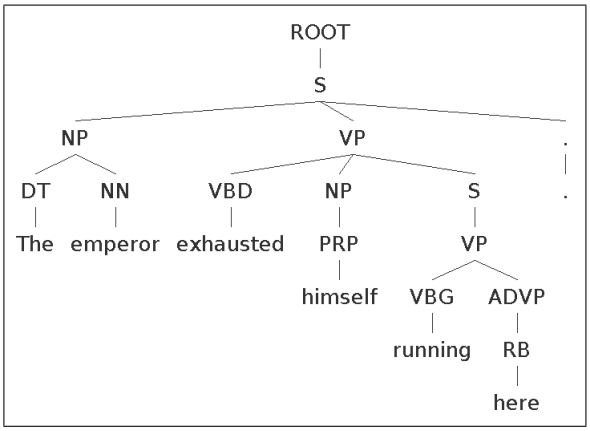












Stanford parses of sentences used (contd.)

Chapter 4

Enhancing WEC by handling special language phenomena

In this chapter, we discuss certain language phenomena that are handled during the WEC Enhancement stage of the UNL Enconversion system. The handling is mainly in the form of minor modifications to the WEC to account for the syntactic differences that are there between UNL and the transfer facts produced by XLE parser.

4.1 Empty Pronominals

This kind of phenomenon can be present in two kinds of sentence, namely in TO-infinitival clauses and Gerundial clauses. The following sections provide a detailed description of the phenomenon followed by the algorithm that is used to tackle this constructs of this kind.

4.1.1 TO-Infinitival clause

Theoretically, to-infinitival clauses have an empty pronominal, called PRO, which is covertly present as the grammatical subject of the clause. To generate the correct UNL expressions, it is important that the correct PRO element is detected and included in the final UNL expression. For example take the following sentence.

I forced him to read the book.

The UNL expression for the above sentence is as follows.

```
[unl:1]
agt(force:2.@entry.@past,I:0)
agt:01(read:9.@entry,he:5)
gol(force:2.@entry.@past,he:5)
obj(force:2.@entry.@past,:01)
obj:01(read:9.@entry,book:13.@def)
[/unl]
```

The PRO element in the above sentence is *he* and it should also be present at the position shown below semantically.

I forced him [[him] to read the book].

It can be seen from the UNL expression that *him* has been correctly identified as the **agt** of *read*. Consider another example.

I promised him to read the book.

The UNL expression for the above sentence is as follows.

```
[unl:1]
agt:01(read:8.@entry,I:0)
aoj(promise:2.@entry.@past,I:0)
gol(promise:2.@entry.@past,he:4)
obj(promise:2.@entry.@past,:01)
obj:01(read:8.@entry,book:12.@def)
[/unl]
```

In this case the correct PRO element is *I* and not *he*. The sentence shown below shows the position of the PRO element.

I promised him [[I] to read the book].

It can be seen from the UNL expression that the word *I* has been correctly identified as the **agt** of *read*. From the two sentences considered, it can be seen that for different verb types, depending upon the semantics of the verb the PRO element is differs. This PRO element handling is done by correctly by the XLE parser. It does this by correctly identifying the subject of *read*. Hence, no special handling is necessary for this case.

4.1.2 Gerundial Clauses

Consider the following sentence.

He slept, crying.

The UNL expression for the sentence is as under.

```
[unl:1]
agt(cry:7.@entry.@progress,he:0)
aoj(sleep:2.@entry.@past,he:0)
[/unl]
```

Here also, just like the TO-infinitival clause, a covert subject for *he* is present. But these kinds of clauses are not handled by the XLE parser. It just inserts a special word `null-pro`as the subject of *cry* which indicates that *cry* contains an empty pronominal. Hence, without any special handling the UNL output of the system would contain the following UNL expression.

```
agt(cry:7.@entry.@progress,null_pro:0)
```

The following algorithm takes care of this situation.

Algorithm

1. The transfer fact that connects the main predicate (say w1) in the subclause to the empty pronominal place holder is noted. (In the above case, it is the transfer fact `SUBJ` that connects the main predicate *cry* in the subclause to `null-pro`).

2. The word entry (say w2) for which the main predicate for the empty pronominal occurs as a value of the transfer fact is retrieved. (In the above case, *cry* appears as the `ADJUNCT` of *sleep*).

3. w2 is checked to see if it has a same transfer fact (say tf) that connected the main predicate in the subclause (w1) to `null-pro`.(In the above case, *sleep* indeed has a `SUBJ` transfer fact connecting it to *he*).

4. If found, the word entry connected by this transfer fact (tf) in w2 is set as the replacement for the empty pronominal of w1. Else recursion takes place with w2, traveling one step backwards.

4.2 Relative Pronouns

A relative pronoun is a pronoun that marks a relative clause within a larger sentence. It is called relative pronoun because it refers to the word that modifies it. For example take the following two sentences.
John has the book. Mary wants the book.
John has the book that Mary wants.

In the above sentence, the relative pronoun *'that'* links the two clauses into a single complex clause. In the second sentence, *'that'* refers to *'the book'*. If this case is not handled properly, then the following UNL expression is generated.

```
[unl:1]
aoj(have:1.@entry.@present,John:0)
aoj(want:10.@entry.@present,Mary:9)
obj(have:1.@entry.@present,book:4.@def)
obj(want:10.@entry.@present,that:5.@def)
[/unl]
```

It is clear that the following relation in the above UNL expression is incorrect.

```
obj(want:10.@entry.@present,that:5.@def)
```

The correct relation for this case should be as follows.

```
obj(want:10.@entry.@present,book:4.@def)
```

The following algorithm takes care of this situation.
Algorithm

- The word entry collection is checked for a relative pronoun. This is done by searching for a `PRON-TYPE` transfer fact with its value as `rel`.
- In case of a relative pronoun (say w) the following steps are performed.
  - The word entry (say wc) for the transfer fact PRON-REL in which w appears as the value of the transfer fact is retrieved.

- The word entry (say wr) is retrieved in which wc appears as the value of the ADJUNCT transfer fact.
- The OBJ chain from the wRet is traversed until the end of the chain is reached (i.e. wRet should not have a further OBJ transfer fact).
- The new wRet replaces the relative pronoun in the UNL relation.

4.3 Multiwords

Mulitiword expression can be of the following types

- Phrasal verbs: *break in, put up, wear out* etc.
- Multiword Prepositions: *out of, according to, because of* etc.
- Multiword nouns: These can be again of two types.
  - Common noun: *fan mail, car pool* etc.
  - Proper nouns: *John Smith, Mr. Bond* etc.

These kind of multiwords are common in English language and hence special handling is required handle such cases as described below.

4.3.1 Phrasal Verbs

XLE parser detects these kind of multiwords correctly and hence there is no additional need for special handling.

4.3.2 Multiword Prepositions

Consider the following sentence.
John looked into the bag out of curiosity.

In the above sentence, *'out of'* is a prepositional multiword. But XLE parser cannot identify this multiword. What it does instead is that it considers *'out'* as an ADJUNCT of *'bag'*, *'of'* as an

OBJ of *'out'* and *'curiosity'* as an OBJ of *'of'*. But as *'out of'* is a multiword expression, what is really needed is *'out of'* as an ADJUNCT of *'bag'* and *'curiosity'* as an OBJ of *'out of'*.

We can see here, the chain of OBJ transfer facts begin at the first word of the preposition *'out'* and end at the final object of the complete preposition *'curiosity'*. All the words from the first word up to and excluding the final object need to be clubbed together to form the multiword preposition. We call this process object chaining.

All the prepositions of a sentence will either contain a PTYPE or a PSEM transfer fact in the WEC. Whenever a preposition is encountered, the following algorithm is employed to check for a preposition multiword.

Algorithm

1. Let w be the preposition.
2. A new lexical item wMulti is defined as an empty string.
3. While w has an OBJ transfer fact defined on it, w is appended to wMulti.
4. A new word enrty is created for wMulti in the Word Entry Collection.
5. This new word entry behaves in the same way as the last link in the OBJ chain. Hence, all its features and transfer facts are copied into wMulti.
6. wMulti is the new multiword in place of the object chain.

4.3.3 Multiword Nouns

(a) Common Nouns Consider the following sentence. *The actor receives huge fan mail.*

Here, *fan* is the mod of mail. So the UNL expression is

```
[unl:1]
agt(receive:2.@entry.@present,actor:0)
mod(mail:5,huge:3)
mod(mail:5,fan:4)
obj(receive:2.@entry.@present,mail:5)
[/unl]
```

But *fan mail* is a multiword and its correct representation should be as under

```
[unl:1]
agt(receive:2.@entry.@present,actor:0)
mod(fan_mail:1001,huge:3)
obj(receive:2.@entry.@present,fan_mail:1001)
[/unl]
```

But not all cases of `MOD` form a multiword. Hence, there is no way to know which `MOD` transfer facts give rise to multiwords. We prepared a database of all multiwords using WordNet 3.0. The following algorithm is used to check for multiwords.

Algorithm

1. All the word entries that contain a `MOD` transfer fact are retrieved.
2. For each word entry (say w) in this list, the following steps are performed.
   (a) A multiword (say wMulti) is constructed by concatenating with the target word entry for the `MOD` transfer fact of w.
   (b) This multiword is checked for existence in the database.
   (c) If it exists the following steps are performed.
      i. A new word entry is created corresponding to this multiword
      ii. All occurrences of w are replaced with wMulti.
      iii. The word entry that was merged with w is deleted along with the associated features.

(b) Proper Nouns We identify proper nouns using the NER system. If the propernoun is a multi-word, we handle it the same way as common nouns except that there is no database lookup.

4.4 Cleft Sentences

A cleft sentence is a sentence formed by a main clause and a subordinate clause, which together express a meaning that could be expressed by a simple sentence. Clefts typically put a particular

constituent into focus. For example consider the following sentence.

It was John who broke the window.

This sentence has the same meaning as the following sentence but with an emphasis on *John*.

John broke the window.

XLE parser indicates this phenomenon by `TOPIC-CLEFT` transfer fact. The following algorithm describes the process of handling these constructs in detail.

Algorithm

1. All the word entries that contain a `TOPIC-CLEFT` transfer fact are retrieved. This transfer fact essentially indicates the topic of the sentence in the cleft-construction and also connects it to the main predicate. (In the above sentence *'John'* has a `TOPIC-CLEFT`transfer fact and it connects it to the main predicate *'break'*).

2. For each of these words (say wCleft) the following steps are performed.

   (a) The target word (v) of the TOPIC-CLEFT transfer fact is retrieved.

   (b) All the transfer facts that this v participates in are retrieved.

      i. If any of these transfer facts contains `PRON-TYPE:rel` transfer fact, then the value of v is replaced with w, i.e. the word entry of TOPIC-CLEFT.

4.5 There Construction

Consider the following sentence.

There are many people in this room.

This sentence can also be written in the following form.

Many people exist in this room.

The UNL expression for both of the above sentences should be the same and is as shown below.

```
[unl:1]
aoj(exist:3.@entry.@present,people:2.@pl)
qua(people:2.@pl,many:0)
```

```
scn(exist:3.@entry.@present,room:6)
[/unl]
```

Thus, some special processing is required to generate the correct UNL expression for the first sentence. The following algorithm illustrates the technique.

Algorithm

1. The subcategorization frame `V-SUBJexpl-XCOMPRED-there` indicates the presence of *there* constructions in the sentence. It is checked if any of the word entries (say w) contain this transfer fact.

2. For w, a new word entry *'exist'* is created. Its `SUBJ` is set as `XCOM-PRED` of the word entry w. `TENSE` of this new word entry *'exist'* is set the same as that of w. Its `VTYPE` is set to main and the `SUBCAT-FRAME` is set as `V-SUBJexpl-XCOMPRED-there`.

4.6 Handling *weekday*

The XLE parser, when producing transfer facts for the names of weekdays, converts them to a special notation. For instance, when producing the transfer facts for *Monday*, it converts it to `weekday(Monday)`. Thus, when the UNL expressions are produced from this transfer fact, though they contain the correct words and relations, since the universal words are incorrect, the UNL expression becomes incorrect. To avoid this, we replace `weekday(Monday)` by `Monday`.

4.7 Gerund stemming

The XLE parser produces stemmed words in its transfer facts. While this is good as the final UNL expression should also contain stemmed words, stemming is not needed all the time. For instance, the XLE parser cannot distinguish between gerunds and normal verbs. Take the example of the following sentence.

His writing brings him Rs 10000 a year.

Here *'writing'* is a gerund. But the XLE parser cannot identify this. Thus while producing the transfer facts it stems *'writing'* to *'write'*. Though the final UNL expression that gets pro-

duced is correct, since the universal word is wrongly stemmed, the UNL expression becomes incorrect.

To solve this problem, the Stanford Parser is used. The POS tag of the Stanford Parser is used to check whether the word is a gerund. The suffixes of the words are also checked to confirm the information retrieved from the Stanford Parser. Once found to be a gerund, wherever the word occurs in the transfer facts, the stemmed word is changed to the original word.

4.8 Degrees of comparison

When the XLE parser comes across comparative or superlative adjectives like *longer*, *later* etc., it breaks them down. For instance when it comes across *tallest*, it breaks it down to two words, *'tall'* and *'most'*. Though this is not wrong, it creates a problem generating the correct UNL expression. Consider the following sentence.
He is the tallest boy.

Some of the transfer facts that get produces are as follows.

```
ADJUNCT, boy, tall
ADJUNCT, tall, most
ADJUNCT-TYPE, most, degree
```

Thus to produce the correct result we convert the above transfer facts into `ADJUNCT, boy, tallest`. And we remove the `ADJUNCT` transfer fact between *'tall'* and *'most'*. We also replace all occurrences of *'tall'* by *'tallest'*.

4.9 Modals

When modals appear in the sentence, sometimes the XLE parser marks them as the main verb and generates transfer facts with respect to them instead of considering the actual main verb. Consider the following sentence.
I know he can swim.

In the above sentence, the complement of the verb *'know'* is *'swim'*, but instead, the XLE parser makes *'can'* the complement of *'know'*. This creates a problem when generating the `obj` relation between *'know'* and *'swim'*. The following algorithm is used to tide over this situation.

Algorithm:

1. All the word entries are searched for modal verbs. This can be done by checking whether the word entry has a transfer fact containing the transfer fact `VTYPE` with its value as 'modal'.

2. For the modal verb, its main verb is identified. This is done by checking the value of the `XCOMP` transfer fact. The value of the `XCOMP` transfer fact of the modal verb is always its main verb.

3. Once the main verb is identified, all the transfer facts where the modal verb occurs as the value is replaced by the main verb.

This corrects the problem since, at the time of relation generation, the system will come across the transfer fact containing the main verb instead of the modal verb.

4.10 Summary

In this chapter, we described the processing we perform on the WEC to make it more amenable to UNL generation. This concludes the discussion of our Enconverter system. In the next chapter, we discuss a few problems with the system, ways to handle them and report the improvements in terms of precision, recall and F-score.

Chapter 5

Error Analysis and Enhancements

We studied the generated UNL graphs and Gold UNL of around 30 sentences with an average length of 20 words from the UNL EOLSS corpus[EOL10]. We have discovered a number of errors and implemented solutions for the same. This chapter goes into the details of these errors and how we resolved them. We first discuss the enhancements we made to the knowledge and rule bases and then move on to discuss enhancements to the system logic.

5.1 Knowledge and Rule Base Enhancements

In this section, I discuss the problems which were solved by improving our knowledge and rule bases.

5.1.1 *aoj-mod* confusion

For many sentences with attributive adjectives, we were generating `mod` instead of `aoj`. For example, consider the sentence

The toxic effects of water pollution are classified into acute effects and long-term chronic effects.

The actual and generated UNL graphs are shown in Figure 5.1. The errors in the generated UNL are shown in red.

Solution

We solved this problem by modifying a rule in the Adjunct relation rules table (Table 3.11, Page 23). The modification is shown in Table 5.1.

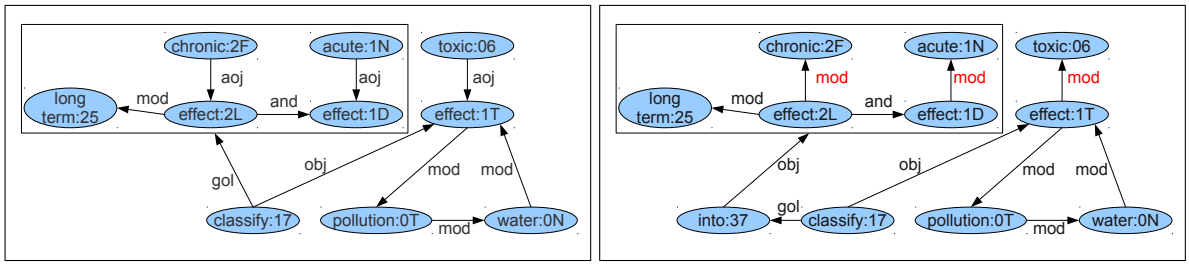


Figure 5.1: aoj - mod confusion; Left - Gold UNL, Right - Generated UNL

Table 5.2: Improvement after solving the *aoj-mod* confusion

	Sentence length	Precision	Recall	F-Score
Before	> 12 and < 25	36.54	29.61	33.19
After		39.84	32.13	35.42
Before	> 25	27.71	22.55	24.91
After		31.81	25.89	28.32
Before	> 35	22.66	18.70	20.20
After		26.50	21.69	23.54

Table 5.1: Solving the aoj-mod confusion

	transfer_fact_1	**w2_prop**	**w2_features**	**relation**	**uw1**	**uw2**	**attrib**
Before	ADJUNCT	ATYPE=attributive	Null	mod	1	2	Null
Now	ADJUNCT	ATYPE=attributive	Null	aoj	2	1	Null

Results

For testing our system, we randomly selected sentences of various lengths from the UNDL EOSS corpus. In our test set, we have about 200 sentences of lengths 12-25 words, longer than 25 words, and longer than 35 words. Table 5.2 shows the results[1] on our test set before solving the problem and after solving it.

5.1.2 *obj-aoj* confusion

For many sentences, we generate `aoj` relation between verbs and their objects instead of generating `obj`. For example, consider the sentence

The use of treated sewage water for irrigating crops is accepted practice in countries such as Egypt, Israel, Jordan, Morocco and Tunisia.

[1]The calculation methodology is described in 8.1

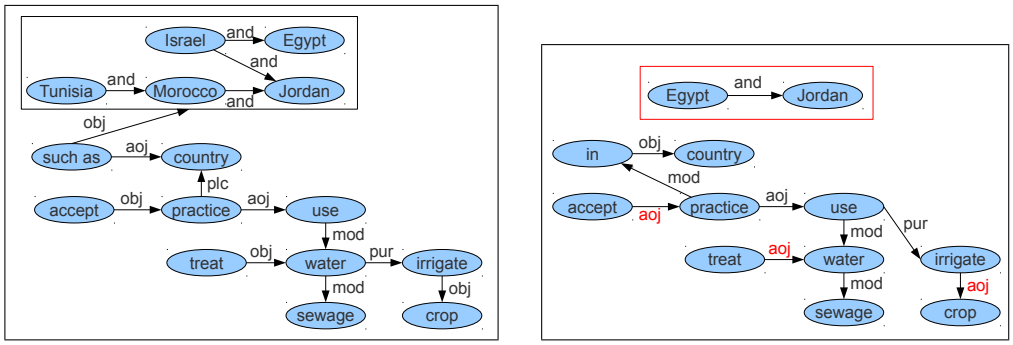


Figure 5.2: obj - aoj confusion; Left - Gold UNL, Right - Generated UNL

Table 5.4: Improvement after solving *obj-aoj* confusion

	Sentence length	Precision	Recall	F-Score
Before	> 12 and < 25	39.84	32.13	35.42
After		40.37	32.60	35.92
Before	> 25	31.81	25.89	28.32
After		32.48	26.39	28.89
Before	> 35	26.50	21.69	23.54
After		27.03	22.06	23.97

The actual and generated UNL graphs are shown in Figure 5.2. The errors in the generated UNL are shown in red.

Solution

We solved this problem by modifying a rule in the Adjunct relation rules table (Table 3.11, Page 23). The modification is shown in Table 5.3

Table 5.3: Solving the obj-aoj confusion

	transfer_fact_1	**w2_prop**	**w2_features**	**relation**	**uw1**	**uw2**	**attrib**
Before	ADJUNCT	DEVERBAL=prog DEVERBAL=pass	Null	aoj	1	2	Null
Now	ADJUNCT	DEVERBAL=prog DEVERBAL=pass	Null	obj	2	1	Null

Results

The results of this modification on our test set are shown in Table 5.4. The 'Before' row gives the results of the modification from the previous subsection (Section 5.1.1). The 'After' row gives result after solving the *obj-aoj* problem.

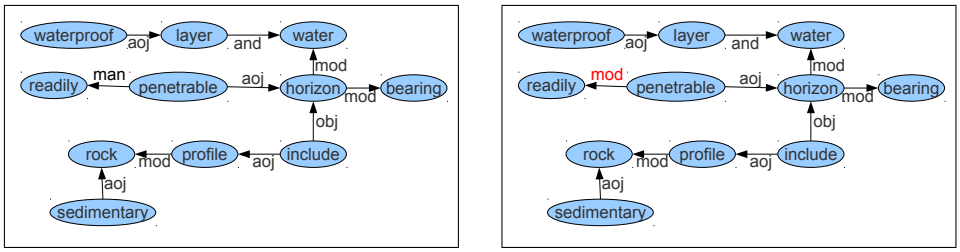


Figure 5.3: Error generating man; Left - Gold UNL, Right - Generated UNL

Table 5.5: Improvement after solving the *man, ben, gol* problem

	Sentence length	Precision	Recall	F-Score
Before	> 12 and < 25	40.37	32.60	0.35.92
After		40.77	32.97	36.30
Before	> 25	32.48	26.39	28.89
After		32.82	26.52	29.18
Before	> 35	27.03	22.06	23.97
After		27.18	22.18	24.10

5.1.3 *ben, gol* and *man*

In some sentences, we were not detecting adverbs of manner correctly and hence were not generating `man` relation correctly. For example, consider the sentence

The profile of sedimentary rocks includes water-bearing horizons, which are readily penetrable for water, and waterproof layers.

The actual and generated UNL graphs are shown in Figure 5.3. The errors in the generated UNL are shown in red.

Another problem is the confusion between `ben` and `gol` relations. Since `ben` and `gol` are close relations, it is sometimes hard to decide which to use.

Solution

We changed the classes of all verbs ending with *'ly'* in the Adverb class table (Table 3.5, Page 19) to *MANNER*. Also, to deal with the `ben-gol` confusion, we created a list of 305 verbs and the relations (`ben` or `gol`) associated with them.

Results

The results on our test set before and after solving this problem are in Table 5.5.

5.1.4 Verb type determination

The system used a few heuristics to determine the verb type (do/be/occur). Since this does not always yield the correct type, we created a list of 3734 verbs with their types. The list contains only those verbs which have a single type in all contexts and usages. A few sample entries are shown in Table 5.6.

Table 5.6: Verb - Type mapping

verb	**type**
abandon	occur
abbreviate	do
abduct	do
abhor	be
abide_by	do
ablactate	do

Results

The results on our test set before and after solving this problem are in Table 5.7.

Table 5.7: Improvement after fixing the verb type problem

	Sentence length	Precision	Recall	F-Score
Before	> 12 and < 25	40.77	32.97	36.30
After		41.23	33.25	36.65
Before	> 25	32.82	26.52	29.18
After		33.10	26.89	29.44
Before	> 35	27.18	22.180	24.10
After		27.33	22.29	24.23

5.2 System Enhancements

The modifications and additions discussed in this section are the changes that were made to the system rather than to the knowledge/rule bases (which was the topic of the previous section).

5.2.1 Handling comma separated lists

For sentences which contain comma separated lists, we observed that all members of the list were not generated. For example, consider the sentence
The use of treated sewage water for irrigating crops is accepted practice in countries such as Egypt, Israel, Jordan, Morocco and Tunisia.

The actual and generated UNL graphs are shown in Figure 5.2, Page 49. The errors in the generated UNL are shown in red box. We see that only *'Egypt'* and *'Jordan'* are generated and the remaining items of the list are missing.

Solution

For the above sentence, the XLE transfer facts generated for the comma separated list are

```
COORD,coord_and_Egypt_Jordan_Tunisia:36,+_
COORD-ELEM,coord_and_Egypt_Jordan_Tunisia:36,Egypt:158
COORD-ELEM,coord_and_Egypt_Jordan_Tunisia:36,Jordan:160
COORD-ELEM,coord_and_Egypt_Jordan_Tunisia:36,Tunisia:163
COORD-FORM,coord_and_Egypt_Jordan_Tunisia:36,and
COORD-LEVEL,coord_and_Egypt_Jordan_Tunisia:36,NP
```

We use these to generate the following relations

```
and(jordan(equ>jordan):11,egypt(equ>egypt):10)
and(tunisia(equ>tunisia):12,jordan(equ>jordan):11)
```

Since XLE parser misses Morocco and Israel, they do not feature in the UNL either.

Results

The results on our test set before and after solving this problem are in Table 5.8.

5.2.2 Handling named entities

Just like we assigned classes to common nouns (see Section 3.2.1), it is desirable to assign classes to proper nouns as well. To do this, we performed Named Entity Recognition on

Table 5.8: Improvement after solving the problem with comma separated lists

	Sentence length	Precision	Recall	F-Score
Before	> 12 and < 25	41.23	33.25	36.65
After		42.58	38.02	39.82
Before	> 25	33.10	26.89	29.44
After		33.73	30.65	31.95
Before	> 35	27.33	22.29	24.23
After		27.86	25.20	26.17

Table 5.9: Improvement after adding NER based features

	Sentence length	Precision	Recall	F-Score
Before	> 12 and < 25	42.58	38.02	39.82
After		42.85	38.11	39.97
Before	> 25	33.73	30.65	31.95
After		33.47	30.54	31.72
Before	> 35	27.86	25.20	26.17
After		28.04	25.30	26.25

the sentence. We assigned the class *noun.location* for places, *noun.person* for people and *noun.location, noun.group* for organizations. We then get the additional features based on noun classes in the same way as that for common nouns. Multi word named entities are handled as explained in Section 3.1.2.

Results

The results are in Table 5.9.

5.2.3 Handling null-pro

We introduced the problem due to `null_pro` in section 4.1.2. The solution discussed there does not find a substitute for `null_pro` every time. So, we if a `null_pro` is found in as a feature of any word, we try to determine the actual word using the dependency parse of the stanford parser. Consider the following example -
There lived a king in Delhi.

The XLE output for this sentence contains

```
SUBJ,king:12,null_pro:6
SUBJ,live:1,null_pro:6
```

OBJ,in:14,Delhi:15

There is nothing connecting king:12, live:1 and Delhi:15, and hence UNL generated is incorrect. Since the XLE parser does not provide us the word related to king:12 , we have to get this information from somewhere else. So, we use the stanford dependency parse to get the word associated with *live* and *king* and substitute null_pro with it.

```
expl(lived-2, There-1)
det(king-4, a-3)
dobj(lived-2, king-4)
prep_in(lived-2, Delhi-6)
```

In this example, we replace null_pro:6 with king:12 as derived from dobj(lived-2, king-4).

5.2.4 Word sense disambiguation for better feature generation

Previously, WSD was used only for generating correct universal words. We have now extended its use for feature generation as well. There is ambiguity in selecting noun classes if words are not disambiguated. For example, consider the following sentences and the UNLs generated.

- The president smokes a cigar in his house.
  plc(smoke.@entry.@present,house).
- The president smokes a cigar in his dreams.
  scn(smoke.@entry.@present,dream.@pl).
- The president smokes a cigar in his office.
  scn(smoke.@entry.@present,office).
- The president smokes a cigar in his library.
  man(smoke.@entry.@present,library).

The UNLs for the first two sentences are correct. But for the last two, scn and man are generated instead of plc. This is because the words house, dream, office, library are not disambiguated and all of their senses are considered to determine noun class. For example, for

`office`, all of its classes are considered for feature generation. The noun classes of office and their associated features from our knowledge base (Tables 3.1 and 3.2, Page 17) are

1. noun.act - *EVENT, ABSTRACT*
2. noun.artifact - *ARTIFACT, CONCRETE*
3. noun.group - *SOCIAL_GRP*
4. noun.location - *PLACE*
5. noun.state - *ABSTRACT*

Taking all these features, the relations generated using Table 3.12 (Page 24) are -

- man
- scn
- plc

But we want only the relation `plc` because the sense in use here is (4) - noun.location. Hence, we perform WSD, get the sense ID. We then use this sense ID to determine which lexicographer file[2] it occurs in. The name of the lexicographer file gives us the noun class.

Results

The results are in Table 5.10. Although, ideally, the scores should have increased for all sentences, there is a decrease in scores for some sentences because, as of now, the WSD system is trained on Tourism domain, whereas our corpus is from an Encyclopaedia of Water.

5.3 Dealing with long sentences

To a large extent, we depend on the output of XLE parser to generate UNL. But the failure rate of XLE parser increases as the length of sentences increases. Table 5.11 gives the failure rate of XLE parser on our corpus. To deal with this, we implemented a text simplifier (see Chapter

[2]http://wordnet.princeton.edu/man/lexnames.5WN.html

Table 5.10: Improvement after filtering features based on word sense

	Sentence length	Precision	Recall	F-Score
Before	>12 and <25	42.85	38.11	39.97
After		43.16	38.50	40.32
Before	>25	33.47	30.54	31.72
After		33.44	30.66	31.76
Before	>35	28.04	25.30	26.25
After		27.77	25.15	26.07

Table 5.11: Failure rate of XLE parser

Sentence length	No. of sentences	No. of XLE failures
12 to 25	216	11
>25	213	18
>35	191	46

6) and UNL merger (Chapter 7) and used it as shown in Figure 5.4. The simplifier breaks down sentences into smaller sentences, whose UNLs are generated and then merged to give the UNL of the actual sentence.

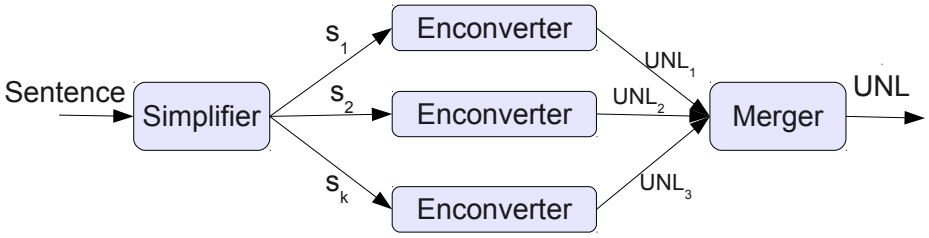


Figure 5.4: Enconverter with pre-simplification of sentence

The results are in Table 5.12. We see that the score reduces for sentences of length less than 25 words, but increases for sentences longer than 25 words. Hence, simplification of long sentences (longer than 25 words) helps generation of UNL.

5.4 Summary

In this chapter, we described how we improved the existing system by making various knowledge and rule base changes and also by making changes to the system logic. In the next chapter, we describe the Text Simplifier we implemented and then we describe our merger.

Table 5.12: Improvement due to pre-simplification and merging

	Sentence length	Precision	Recall	F-Score
Before	>12 and <25	43.16	38.50	40.32
After		41.61	36.23	38.58
Before	>25	33.44	30.66	31.76
After		35.17	31.06	32.69
Before	>35	27.77	25.15	26.07
After		32.02	27.48	29.35

Chapter 6

Text simplification

Text Simplification[Sid02] is a series of syntactic transformations which aim to (i) reduce syntactic complexity (ii) preserve meaning of text. We have built a text simplifier largely based on ideas from [Sid04]. An example of text simplification is shown below

Sentence: *A former ceremonial officer, who was at the heart of Whitehall patronage machinery, said there should be a review of the honours list.*

Simplified: *A former ceremonial officer said there should be a review of the honours list. This former ceremonial officer was at the heart of the Whitehall patronage machinery.*

Figure 6.1 illustrates the 3 stages of the Text simplifier: *Analysis, Transformation* and *Reordering.*

6.1 Analysis

The analysis stage annotates tokens in a sentence with

1. Part of speech tags
2. Chunk tags
3. Grammatical function (subj/obj/iobj/obliq)
4. Agreement features (gender, animacy and number) of noun phrases.
5. Pronoun and anaphora references

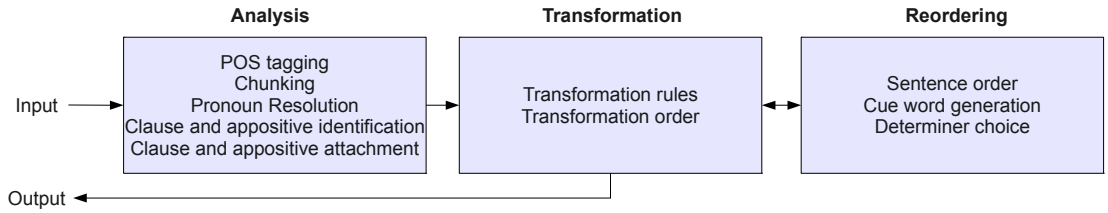


Figure 6.1: The architecture of Text Simplifier (adopted from [Sid04])

6. Marking clause boundaries

7. Appositives[1] and their attachment

The output of the *Analysis* stage is a sentence annotated with the syntactic and semantic information mentioned above. An example annotation for the sentence *'The cathode in the lamp contains the element that should be determined in the analyzed material.'* is shown in Figure 6.2.

We use readily available tools for generating POS tags and marking chunks. We obtain the number from POS tags and the gender and animacy information from WordNet (if available). The grammatical function is marked using regular expressions on chunk tags. The agreement features and grammatical functions are used for resolving pronouns and anaphora. We built the anaphora resolution system based on [Sid03].

We discuss marking clause and appositive boundaries in next.

6.1.1 Marking clause boundaries

We use the following algorithm from [Sid04].

1. `IF` the relative pronoun is immediately followed by a comma `THEN` Jump to the token after the next comma

2. Jump forwards past one verb group and one noun group.

3. `IF`a complementiser was encountered after the verb group, or the verb group contained a saying verb `THEN` jump ahead past the next verb group as well.

4. `FOR` each comma, colon, semicolon, verb group or relative pronoun (processing in a left to right order) `DO`.

[1] An appositive is anything enclosed in commas, not starting with a conjunction and not containing a verb.

```
<s id="0">
        <NP animacy="i" clause="0_8" gender="n" gfunction="subj" id="p_0_0" npid="np_0_0"
        number="s" salience="160.0" type="common">
                <DT id="0_0">The</DT>
                <NN id="0_1">cathode</NN>
        </NP>
        <PP id="p_0_1">
                <IN id="0_2">in</IN>
        </PP>
        <NP animacy="i" gender="n" gfunction="obliq" id="p_0_2" npid="np_0_1"
        number="s" salience="120.0" type="common">
                <DT id="0_3">the</DT>
                <NN id="0_4">lamp</NN>
        </NP>
        <VP id="p_0_3">
                <VBZ id="0_5">contains</VBZ>
        </VP>
        <NP animacy="i" gender="n" gfunction="dobj" id="p_0_4" npid="np_0_2"
        number="s" salience="130.0" type="common">
                <DT id="0_6">the</DT>
                <NN id="0_7">element</NN>
        </NP>
        <NP animacy="a,i" gender="m,f,n" gfunction="iobj" id="p_0_5" npid="np_0_3"
        number="s" pronoun="that" refersTo="np_0_2" rel-clause="0_8" relclause-type="restr"
        salience="40.0" type="pronoun-relative">
                <WDT id="0_8">that</WDT>
        </NP>
        <VP id="p_0_6">
                <MD id="0_9">should</MD>
                <VB id="0_10">be</VB>
                <VBN id="0_11">determined</VBN>
        </VP>
        <PP id="p_0_7">
                <IN id="0_12">in</IN>
        </PP>
        <NP animacy="a" gender="m,f" gfunction="obliq" id="p_0_8" npid="np_0_4" number="s"
        salience="120.0" type="common">
                <DT id="0_13">the</DT>
                <JJ id="0_14">analyzed</JJ>
                <NN id="0_15">material</NN>
        </NP>
</s>
```

Figure 6.2: The output of the analysis stage

(a) IF colon or semicolon or end of enclosing clause, THEN END CLAUSE

(b) IF a comma followed by an appositive (appositive determination is described in section 6.1.2) THEN INTERNAL comma

(c) IF a comma followed by a verb group THEN .

   i. IF the verb has POS "VBN|G" THEN INTERNAL comma

(d) IF a comma that is an implicit conjunction of adjectives or adverbs like "JJ, JJ" or "RB, RB" THEN INTERNAL clause

(e) IF we are inside a Pronoun X noun $X = who|which|that$ THEN.

   i. IF "CC Pronoun X" THEN INTERNAL clause and DELETE "Pronoun X"

   ii. IF ", who|which|that" THEN INTERNAL comma

   iii. Recursively find the end of the embedded clause

(f) `IF` previous token is a conjunction or a subject pronoun (I, they, he, we, she) `THEN` INTERNAL comma

(g) `ELSE` by default end clause

6.1.2 Marking appositives

The parts of sentences which match the following pattern are marked as appositives

, NP ['Prep NP']* [RC]? [,|EOS][2]

Anything that appears between a pair of commans, contains an NP, zero or more occurences of prepositional NPs and zero or more relative clauses is marked as an appositive. For example, *Larry Birns ,* [$_{appositive}$ *director of the Washington-based Council on Hemispheric Affairs , a liberal research group*]*, said that Latin American countries would be profoundly disappointed if Canada were to follow the U.S. lead in the OAS.*

Appositive phrase attachment is decided in the same manner as relative clause attachment (anaphora resolution). The head noun phrase in the appositive is resolved to the most salient noun phrase that agrees with it in number, animacy and gender, subject to a restrictive filter which ensures that the noun phrase that an appositive attaches to can only be separated from it by prepositional phrases.

6.2 Transformation

This stage contains hand crafted rules (from [Sid04]) to break down the sentence into smaller sentences based on the clause information obtained in the analysis stage.

6.2.1 Rules for conjoined clauses

In the following rules, **X** and **Y** represent clauses contained in the sentence.

i. **CC X, Y $\rightarrow$ X. Y.**

where CC matches one of *though, although, when* or *because*

Example: *[Although]$_{CC}$ [both India and Pakistan announced troop withdrawals along*

[2]Standard regular expression syntax is used here

the border]$_{CLAUSE}$, [they both left their forces in Kashmir intact.]$_{CLAUSE}$

Simplification: *Both India and Pakistan announced troop withdrawals along the borders. But*[3] *they both left their forces in Kashmir intact.*

ii. **If X [then |,] Y → X. Y.**[4]

Example: *[If]$_{CC}$ [people have got in place proper effective safety measures]$_{CLAUSE}$, [then]$_{CC}$ [naturally we are pleased about that]$_{CLAUSE}$.*

Simplification: *Suppose people have got in place proper effective safety measures. Then naturally we are pleased about that.*

iii. **X [,]? CC Y → X. Y.**

where *CC* matches one of *though, although, but, and, because, since, as, before, after* or *when*.

Example: *[I have been involved with badgers for 24 years]$_{CLAUSE}$ [and]$_{CC}$ [I have never heard of anything like this]$_{CLAUSE}$.*

Simplification: *I have been involved with badgers for 24 years. And I have never heard of anything like this.*

6.2.2 Rule for relative clauses

W$_x$ X [RC RELPR$_x$ Y] Z →V W X Z. W Y.

x indicates that the relative pronoun RELPR should be referring to W for this rule to fire.

Example: *'The pace of life was slower in those days,' says [51-year-old Cathy Tinsall]$_W$ [from South London]$_X$, [RC who$_{RELPR}$ [had five children, three of them boys]$_Y$].*

Simplification: *'The pace of life was slower in those days,' says 51-year-old Cathy Tinsall from South London. Cathy Tinsall had five children, three of them boys.*

6.2.3 Rule for appositives

U V$_{NP_x}$ W, [APPOS X$_x$ Y], Z → U V W Z. V aux X Y.

Example: *[Pierre Vinken]$_V$, [61 years old]$_{APPOS}$, [will join the board as a non-executive director on Nov 29.]$_Y$*

[3]The cue word 'but' is added by the reordering stage

[4]We use the standard regular expression syntax here, | standing for OR and ? standing for 'zero or one match'

Simplification: *Pierre Vinken will join the board as a non-executive director on Nov 29. Pierre Vinken is 61 years old.*

6.3 Reordering

The ordering of sentences is decided based on the rules which are used in the transformation stage and the rhetorical relation (see [Sid04]) that holds between the resulting pairs of sentences. The cue words and determiners to be added are decided on the basis of rhetorical relation, the tense of the verb and number of the noun phrases involved. Example (1) illustrates reordering and (2) shows the addition of cue word *"This"*.

1. *Mr Anthony, who runs an employment agency, decries program trading, but he is not sure it should be strictly regulated.*
   Correct: *Mr. Anthony runs an employment agency. Mr. Anthony decries program trading. But he is not sure it should be strictly regulated.*
   Wrong: *Mr. Anthony decries program trading. Mr. Anthony runs an employment agency. But he is not sure it should be strictly regulated.*

2. *The man who had brought it in for an estimate returned to collect it.* Here, *it* wrongly refers to *employment agency* instead of *program trading*.
   Simplification: *A man had brought it in for an estimate. This man returned to collect it.*

For the algorithm and other details, please refer to Chapter 5 in [Sid04].

6.4 Summary

In this chapter, we described how a sentence is broken down into smaller sentences. In the next chapter, we describe how we stitch together the UNL graphs generated for each of these smaller sentences.

Chapter 7

Merging UNL graphs

We have simplified a long sentence into smaller sentences and generated UNL graphs for them so far. In this chapter, we see how these UNL graphs can be stitched together to give the UNL graph of the original sentence with minimal loss.

7.1 Merging techniques

While merging the UNL graphs of two sentences (say A and B), we follow one of the below listed techniques.

Edge addition We add a new relation between a UW in UNL_A and a UW in UNL_B.

Node merging A UW in UNL_A and a UW in UNL_B refer to the same entity here. We ensure that both these UWs get the same UW id.

Forest creation We simply append the serialized UNL generated for A to that of B.

Each of the above techniques may also involve deletion of certain nodes (UWs) or edges (relations). As a rule of thumb, we perform *Edge addition* for compound sentences and *Node merging* or *Forest creation* for complex sentences.

7.2 Merging algorithm

The technique adopted for merging depends on the kind of simplification that occurs. We discuss the method for merging graphs of various kinds of simplifications here.

Input: Sentences – A, B; their UNLs – UNL_A, UNL_B.

Output: Merged UNL – UNL_{AB}

7.2.1 Merging for coordinating conjunctions

Consider the following sentence -

A voltmeter is utilized for measuring voltage and an ammeter is used for measuring current.

This sentence is simplified to

A: *A voltmeter is utilized for measuring voltage.*

B: *And an ammeter is used for measuring current.*

The generated UNLs are

```
[unl:A]
obj(utilize:11.@entry.@present,voltmeter:10.@indef)
obj(measure:13.@progress,voltage:12)
rsn(utilize:11.@entry.@present,measure:13.@progress)
[/unl]

[unl:B]
man(use:11.@entry.@present,and:10)
obj(use:11.@entry.@present,ammeter:12.@indef)
obj(measure:13.@progress,current:12)
rsn(use:11.@entry.@present,measure:13.@progress)
[/unl]
```

Algorithm

1. Resolve all id clashes (Eg: `use:11` in UNL_B is changed to `use:51` because its id (i.e 11) clashes with id of `utilize:11`)

2. Remove bad relations (Here, we remove `man( use:11.@entry.@present, and:10)`)

3. Perform *Edge Addition* between words with `@entry` attribute in UNL_A and UNL_B. Label the edge *and* (Here, we add the relation `and( utilize:11.@entry.@present, use:51.@entry.@present)`)

Thus, merged UNL is

```
[unl]
obj(utilize:11.@entry.@present,voltmeter:10.@indef)
obj(measure:13.@progress,voltage:12)
rsn(utilize:11.@entry.@present,measure:13.@progress)
obj(use:51.@entry.@present,ammeter:52.@indef)
obj(measure:53.@progress,current:54)
rsn(use:51.@entry.@present,measure:53.@progress)
and(utilize:11.@entry.@present,use:51.@entry.@present)
[/unl]
```

The algorithm for *or* is exactly the same, except that the relation added is *or* instead of *and*. For *but*, we add the relation `obj(but,`*x*`)` where *x* is the UW with `@entry` attribute in UNL of sentence B.

7.2.2 Merging for subordinating conjunctions

Consider the following sentence -

It is desirable to carry out the analysis within 4 hours when the solution is blue.

This sentence is simplified to

A: *It is desirable to carry out the analysis within 4 hours.*

B: *This is when the solution is blue.*

The generated UNLs are

```
[unl:A]
dur:01(carry out:11.@entry,within:10)
mod(desirable:12,:01)
obj:01(carry out:11.@entry,analysis:13.@def)
obj:01(within:10,hour:14.@pl)
qua:01(hour:14.@pl,4:15)
[/unl]

[unl:B]
aoj(be:52.@entry.@present,this:50)
```

```
aoj(blue:51.@entry.@present,solution:50.@def)
[/unl]
```

Algorithm

1. Resolve all id clashes
2. Remove bad relations (Here, we remove `aoj( be:12.@entry.@present, this:10)`)
3. Perform *Edge Addition* from the subordinating conjunction in *B* to the word with `@entry` attribute in *B*. Label the edge based on the subordinating conjunction – `obj` for *after* and *before*, `rsn` for *since* and `tim` for *when*.
   (Here, we add the relation `tim( blue:54.@entry.@present, when:90)`)

Thus, merged UNL is

```
[unl]
dur:01(carry out:11.@entry,within:10)
mod(desirable:12,:01)
obj:01(carry out:11.@entry,analysis:13.@def)
obj:01(within:10,hour:14.@pl)
qua:01(hour:14.@pl,4:15)
aoj(blue:51.@entry.@present,solution:50.@def)
tim(blue:54.@entry.@present,when:90)
[/unl]
```

7.2.3 Merging for relative pronouns

Consider the following sentence -

The pumps which are used for recharge should not contain any compounds.

This sentence is simplified to

A: *Some pumps are used for recharge.*

B: *These pumps should not contain any compounds.*

The generated UNLs are

```
[unl:A]
obj(use:11.@entry.@present,pump:10.@pl)
pur(use:11.recharge:13)
qua(pump:10.@pl,some:12)
[/unl]

[unl:B]
aoj(contain:11.@entry.@should.@not.@present,pump:10.@pl)
obj(contain:11.@entry.@should.@not.@present,compound:12.@pl)
qua(compound:12.@pl,any:13)
[/unl]
```

Algorithm

1. Resolve all id clashes
2. Remove bad relations, if any (Here, we remove `qua(pump:10.@pl,some:12)`)
3. Perform attribute correction if needed (Here, we add attribute `@def` to `pump:10`)
4. Perform *Node merging* of UWs common to both *A* and *B*. (Here, we do this by assigning the same id to `pump` in UNL_A and UNL_B)

Thus, merged UNL is

```
[unl]
obj(use:11.@entry.@present,pump:10.@pl)
pur(use:11.recharge:13)
aoj(contain:51.@entry.@should.@not.@present,pump:10.@pl.@def)
obj(contain:51.@entry.@should.@not.@present,compound:52.@pl)
qua(compound:52.@pl,any:13)
[/unl]
```

If the sentence was *The pump which is used for recharge should not contain any compounds*, then it would have been simplified to *A pump is used for recharge. This pump should not contain any compounds.*.

We use the same algorithm here too, except that, in this case, we will have to use step (3) to change attribute of `pump:10` from `@indef` to `@def`.

7.2.4 Merging for conjunctions of reason (*'because', 'so', 'therefore'*)

Consider the following sentence -

The color index of the obtained solution is taken as HTM1, because it contains 500g of pure platinum per 2ml of water.

This sentence is simplified to

A: *It contains 500g of pure platinum per 2ml of water.*

B: *So, the color index of the obtained solution is taken as HTM1.*

The generated UNLs are

```
[unl:A]
agt(contain:11.@entry.@present,it:10)
aoj(pure:13,platinum:12)
mod(gram:14.@pl,platinum:12)
mod(milliliter:16.@pl,water:15)
obj(contain:11.@entry.@present,gram:14.@pl)
per(platinum:12,milliliter:16.@pl)
qua(gram:14.@pl,500:17)
qua(milliliter:16.@pl,2:18)
[/unl]

[unl:B]
man(take:11.@entry.@present,as:10)
man(take:11.@entry.@present,so:12)
mod(index:14.@def,color:13)
mod(index:14.@def,solution:15.@def)
obj(as:10,HTM1:16)
obj(obtain:17,solution:15.@def)
obj(take:11.@entry.@present,index:14.@def)
[/unl]
```

Algorithm

1. Resolve all id clashes

2. Remove bad relations (Here, we remove `man(take:11.@entry.@present,so:12)`)

3. Perform *Edge Addition* from word with `@entry` attribute in UNL_B to the word with `@entry` attribute in UNL_A. Label the edge *rsn*. (Here, we add the relation `rsn( take:51.@entry.@present, contain:11.@entry.@present)`)

Thus, merged UNL is

```
[unl]
agt(contain:11.@entry.@present,it:10)
aoj(pure:13,platinum:12)
mod(gram:14.@pl,platinum:12)
mod(milliliter:16.@pl,water:15)
obj(contain:11.@entry.@present,gram:14.@pl)
per(platinum:12,milliliter:16.@pl)
qua(gram:14.@pl,500:17)
qua(milliliter:16.@pl,2:18)
man(take:51.@entry.@present,as:50)
mod(index:54.@def,color:53)
mod(index:54.@def,solution:55.@def)
obj(as:50,HTM1:56)
obj(obtain:57,solution:55.@def)
obj(take:51.@entry.@present,index:54.@def)
rsn(take:51.@entry.@present,contain:11.@entry.@present)
[/unl]
```

7.2.5 Others

For all other kinds of simplifications, we use the *Forest Creation* technique for merging.

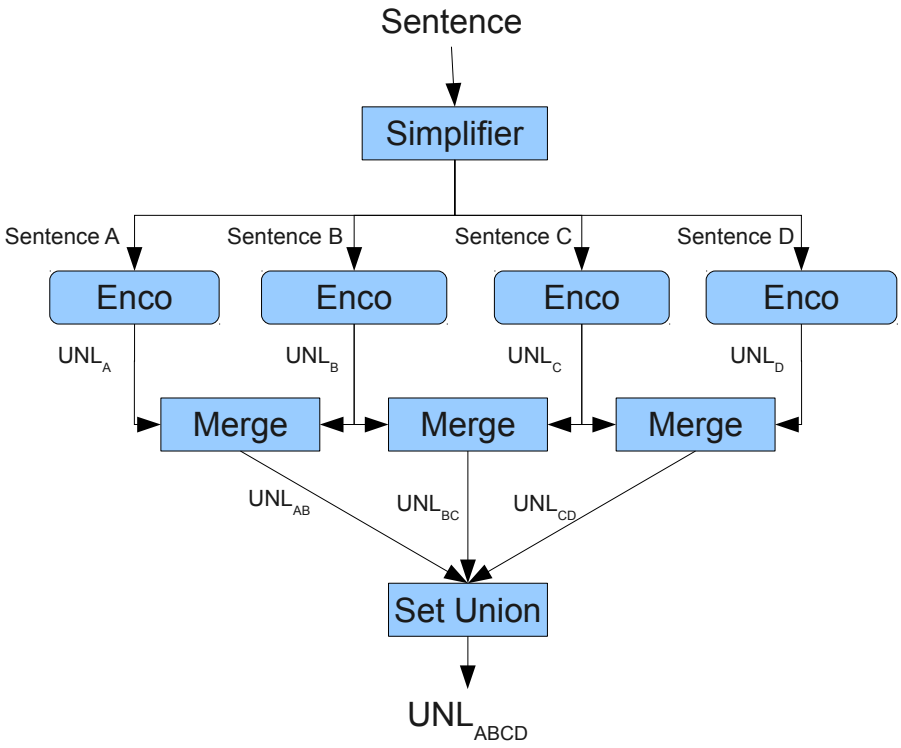


Figure 7.1: Merging more than two sentences

7.3 Adapting the algorithm for merging more than two sentences

There are instances where a sentence is simplified to three or more sentences. When this happens, we merge sentences pair-wise (as discussed in the previous section) and take a set union of UNL expressions ($UNL_{ABC} = UNL_{AB} \cup UNL_{BC}$). This is illustrated in Figure 7.1.

7.4 Summary

In this chapter, we presented our merging strategy. In the next chapter, we present the resulting improvement in UNL generation due to pre-simplification and other enhancements.

Chapter 8

Results

First, we present our evaluation methodology, followed by the results on sentences of various lengths.

8.1 Evaluation methodology

We evaluate the relation generation accuracy and the attribute generation accuracy separately.

8.1.1 Evaluating relations

For evaluating relations, we consider the UNL generated and the Gold UNL (human annotated UNL) for a sentence as a set of triplets. Each triplet is of the form $(relation, uw_1, uw_2)$.

We say triplets $t_{gen} = t_{gold}$ if and only if

1. $relation(t_{gen}) = relation(t_{gold})$
2. $uw_1(t_{gen}) = uw_1(t_{gold})$
3. $uw_2(t_{gen}) = uw_2(t_{gold})$

Table 8.1: Overall improvement of Enconverter

	Sentence length	Precision	Recall	F-Score	Attribute accuracy	Percentage F-score improvement
Then	> 12 and < 25	36.54	29.61	33.19	71.65	
Now		43.16	38.50	40.32	71.18	21.48
Then	> 25	27.71	22.55	24.91	76.42	
Now		35.17	31.06	32.69	78.53	31.23
Then	> 35	22.66	18.70	20.20	69.92	
Now		32.02	27.48	29.35	81.07	45.30

Let $G = \{\text{triplets in the generated UNL}\}$ and $A = \{\text{triplets in the gold UNL}\}$. We define

$$\text{Precision } p = \frac{|G \cap A|}{|G|}$$
$$\text{Recall } r = \frac{|G \cap A|}{|A|}$$
$$\text{F score } f = \frac{2 * p * r}{(p + r)}$$

8.1.2 Evaluating attributes

Whenever $t_{gen} = t_{gold}$, let $a_{match}(t_{gen}, t_{gold}) = \{\text{number of attributes in } t_{gen} \text{ that match the attributes in } t_{gold}\}$. Let $count(t_x) = \{\text{number of attributes in } t_x\}$. For a UNL graph, we define

$$\text{Attribute accuracy } = \frac{\sum_{t_{gold}=t_{gen}, t_{gold} \in A, t_{gen} \in G} a_{match}(t_{gold}, t_{gen})}{\sum_{t_{gold}=t_{gen}} count(t_{gold})}$$

8.2 Evaluation

We ran our Enconverter (with and without pre-simplification) on 200 randomly chosen sentences of specific lengths from the UNL EOLSS corpus [EOL10]. The various improvements we made were discussed in Chapter 5. The accuracy figures of the old system and the current system are shown in Table 8.1.

We see that the largest improvement in F-score has been for longer sentences, which shows the effect of our main contribution - use of text simplifier improves UNL generation. We also

see improvement in attribute generation.

8.3 Software released

We have released the following software for public use -

1. A web based English - UNL - Hindi translator, available at *http://www.cfilt.iitb.ac.in:8080/unlmt*

2. A salience based anaphora resolver, executable binary and source SVN repository available under GPL v2.0 at *http://code.google.com/p/anaphoraresolver/*

3. A text simplifier, executable binary and source SVN repository available under GPL v2.0 at *http://code.google.com/p/textsimplifier/*

Chapter 9

Conclusion and future work

The goal of this work was to improve the English to UNL enconverter. We performed a thorough error analysis and identified several problems with the system. After recognizing the causes, we classified the problems into into 3 types based on the solution we came up with -

1. Problems due to knowledge and rule bases
2. Problems in system logic
3. Problem of long sentences

We solved (1) by making several modifications to the rule bases and enriching the knowledge bases. We solved (2) by making suitable changes to the system and adding external systems like WSD and NER. We solved (3) by implementing a text simplifier for breaking down a long sentence into smaller sentences, generating UNL for each of the smaller sentences and strategically merging the UNL graphs.

We performed several experiments and discovered that it is best to simplify only sentences longer than 25 words. We have improved the F-score of the Enconverter by 21.48% for sentences smaller than 25 words, by 31.23% for sentences longer than 25 words and by 45.3% for sentences longer than 35 words.

We have made the text simplifier and anaphora resolver available under GPL. Also, we have made available a web system for English-UNL-Hindi translation.

But the enconverter system still gives a low F-score of approximately 40 for real world sentences. This makes it infeasible for completely automated UNL generation. But it can be

useful for machine aided UNL graph construction by human annotators. Hence, further error analysis is needed. The rule and knowledge bases have to be improved and expanded. Also, we use a naive merging strategy for complex sentences. This needs to be replaced by a non-trivial merging strategy.

Recently, UNDL made available an English-UNL corpus with approximately 13,000 sentences. We had to resort to a rule based system because of unavailabilty of large annotated corpora. But now, since a large annotated corpus is available, machine learning techniques may be used for UNL generation.

Appendix A

XLE parser terminology

Attribute	Meaning	Values	Meaning	Example
PRON-TYPE	Pronoun type	DEMON	Demonstrative pronoun	this/those etc
		EXPLETIVE	Expletive pronoun[1]	*It* is raining.
				There is an apple on the table
		INT	Interrogative pronoun	
		REL	Relative pronoun	He is the one *who* did it.
		REFL	Reflexive pronoun	himself/herself etc
		FREE	Free relative	whatever/whoever
		PERS	Personal pronoun	he/she etc
PRON-FORM	The pronoun itself			Eg: he/she/who
TNS-ASPECT	Tense and Aspect	PRES	Present	
		PAST	Past	
		FUT	Future	
		PERF=+	Perfect	
		PERF=-		
		PROG=+	Progressive	
		PROG=-		
NTYPE	Noun type	COUNT	Common noun	dog, tractor etc
		MASS	Mass nouns	water
		PROPER	Proper noun	Bangalore
ATYPE	Adjective type	ATTRIBUTIVE	Prenominal adjective[2]	The *barking* dog,
				The *small* cat
		ORDINAL	Shows position	The *third* tractor
		CARDINAL	Number	The *three* dogs
		PREDICATIVE	Adjective is a predicate[3]	The tractor is *hard* to push
				(VP argument) **vs**
				The tractor is *red.* (No argument)
				He is *proud* of the tractor
				vs He is *red* of the tractor
PTYPE	Preposition type	SEM	Semantic preposition	Locative, instrumental
				or directional prepositions
		NOSEM	Non-semantic preposition	I referred *to* the book
PSEM	Semantic preposition type	LOCATIVE		on, in, under
		INSTRUMENTAL		with
		DIRECTIONAL		onto, into

[1] Pronouns which do not refer to an actual entitity

[2] An adjective which occurs before a noun

[3] A predicate has a subcat frame of its own and hence takes arguments

PCASE	The preposition itself	ON, WITH, ONTO etc		
ADV-TYPE	Adverb type	ADJMOD ADVMOD VPMOD SMOD	Modifier of adjective Modifier of adverb VP modifier Clause modifier	The *very* grey dog She did it *gently* She *usually* drives home
ADEGREE	Degree of comparison	COMPARATIVE EQUATIVE SUPERLATIVE	better as good as best	
ADEG-TYPE	Type of comparison	POSITIVE NEGATIVE	more less	
ADJUNCT	Adjunct[4]			
ADJUNCT-REL	An adjunct with a relative pronoun			
ADJUNCT-COMP	An adjuct with a degree of comparison			good, better, best
ADJUCT-PAREN	A parenthetical			The button *(2)* is for the oil filter. A warning light - *red or green* - will come up
COMP	A closed complement with its own subject[5]			I know *that this tractor is red.*
XCOMP	A complement where subject of verb is functionally controlled from outside the clause			The woman wants *to drive* *the tractor.*
COMP-EX	Occurs with bridge verbs.	In the example, think is a bridge verb and whisper. is not		What does Mary think *that John drives* : COMP-EX What does Mary whisper *that John said* : COMP

[4] A grammatical function not subcategorized for by the verb (all adjectives are treated as adjuncts)

[5] A complement occurs after a verb and is essential to complete the sentence

Bibliography

[CDK+08] Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. *XLE documentation*. Palo Alto Research Center (PARC), 2008.

[Dal01] Mary Dalrymple. *Syntax and Semantics*, volume 34, chapter Lexical Functional Grammar. Academic Press, 2001.

[EOL10] Unl eolss corpus. http://www.undlfoundation.org/eolss/corpus.htm, June 2010.

[Fel98] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[Gho08] Avishek Ghosh. Semantics extraction from text. Master's thesis, IIT Bombay, 2008.

[Lim07] Sandeep Limaye. Semantics extraction from text in the unl framework. Master's thesis, IIT Bombay, 2007.

[MMS93] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Sid02] Advaith Siddharthan. An architecture for a text simplification system. In *Language Engineering Conference (LEC'02)*, pages 64–71, 2002.

[Sid03] Advaith Siddharthan. Resolving pronouns robustly: Plumbing the depths of shallowness. In *Workshop on Computational Treatments of Anaphora, 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 7–14, 2003.

[Sid04] Advaith Siddharthan. *Syntactic Simplification and Text Cohesion*. PhD thesis, University of Cambridge, August 2004.

[UND04] UNDL Foundation, UNL Center. *Universal Networking Language specification*, 3rd edition, December 2004.

[UND10] Undl foundation. http://www.undlfoundation.org, June 2010.