A Corpus-based Approach for the Automatic Development of UNL Grammars

Sameh Alansary

Bibliotheca Alexandrina, Alexandria, Egypt
Department of Phonetics and Linguistics, Faculty of Arts, Alexandria University
El Shatby, Alexandria, Egypt
Sameh.alansary@bibalex.org

***Abstract*— This paper aims to present the process of the automatic extraction of the tree-to-tree (TT) rules which constitute the most intricate phase in developing the UNL analysis and generation grammars. During analysis, tree-to-tree rules are used to reveal the deep syntactic structure out of the surface syntactic structure; while in generation, they are used to transform the deep syntactic structure into a surface structure. In this paper, we present a method to extract automatically TT rules using an annotated corpus. The availability of large syntactically annotated corpora such as the Penn Tree Bank presents us with the opportunity to automatically build broad coverage grammars. Experiments have proved that grammars that are dependent on the context can be more effective than context-free grammars. We have automatically constructed the TT rules using the Penn Tree Bank version 2.**

1 INTRODUCTION

In the beginning of 2009 a new system came to light in language processing called the UNL+3; UNL+3 is the latest development to the first generation of UNL (more information about the earlier system can be found at http://www.undl.org/unlsys/unl/unl2005/). UNL+3 offers improvements to the existing infrastructure, however, it changes some of the core fundamentals of UNL. The change encompasses the linguistic components (Universal Words, Relations, Attributes and Features) as well as the non-linguistic ones (tools, engines and applications). Also, the structure of the linguistic resources (grammars, dictionaries, corpora..etc.) that handle these components has been drastically changed (more information about UNL+3 can be found http://www.unlweb.net/wiki/).

Natural language sentences and UNL graphs are supposed to convey the same amount of information in different structures: whereas the former arranges data as an ordered list of words, the latter organizes it as a hyper-graph. In that sense, translating from natural language into UNL and from UNL into natural language is ultimately a matter of transforming lists into networks and vice-versa. The UNDL Foundation's generation and analysis tools; Eugene and IAN, assume that such transformation should be carried out progressively, i.e., through a transitional data structure: the tree, which can be used as an interface between lists and networks. Accordingly, the UNL Grammar utilizes seven different types of rules: LL, TT, NN, LT, TL, TN and NT.

In analysis (NL-UNL), list-to-list or List Processing (LL), list-to-tree or surface-structure Formation (LT), tree-to-tree or syntactic processing (TT), tree-to-network or deep-structure formation (TN) and network-to-network or semantic processing (NN) rules are used. The Natural language original sentence (NL) is supposed to be preprocessed by the LL rules in order to become an ordered list. Next, the resulting list structure is parsed with the LT rules so as to unveil its surface syntactic structure which is syntactic tree. The tree structure is further processed by the TT rules in order to expose its inner organization; the deep syntactic structure which is more suitable to semantic interpretation. Then, this deep syntactic structure is projected into a semantic network by TN rules. The resultant semantic network is then post-edited by the NN rules in order to comply with UNL standards and finally generate the UNL Graph.

In generation, the five types of rules used are the network-to-network or semantic processing (NN), network-to-tree or deep-structure Formation (network-to-tree), tree-to-tree or syntactic processing (TT), tree-to-list or surface-structure formation (TL) and list-to-list or list processing (LL) rules. The UNL graph is preprocessed by the NN rules in order to become a more easily tractable semantic network. The resulting network structure is converted by the NT rules into a syntactic structure that is still distant from the surface structure, since it is directly derived from the semantic arrangement. This deep syntactic structure is subsequently transformed into a surface syntactic structure by the TT rules. The surface syntactic structure undergoes many other changes according to the TL rules, which generates a NL-like list structure. This list structure is finally realized as a natural language sentence by the LL rules.

The tree-to-tree rules (TT) phase is a common phase in both the analysis and generation grammars; they are used for processing trees, both in analysis and in generation. During analysis, these rules are used to reveal the deep structure out of the surface structure; however, in generation, they are used to transform the deep syntactic structure into a surface structure.

The TT is the most challenging phase in the UNL grammar, hence, it was necessary to think about a way to automate it. For this purpose, empirical rather than introspective data is required in order to express the authenticity of the language; a corpus-based approach is selected to be the base of the automatic extraction of the tree-to-tree module.

Corpus-based studies provide a means for handling large amounts of language and keeping track of the contextual factors. The availability of resources such as the Treebank[1] has a great impact on the investigations that depend on this kind of studies, and has paved the way for the automatic updating of grammar which first emerged in the nineties. Moreover, the availability of the Treebank caused us to wonder whether it is possible to extract grammar rules from surface structures using the Treebank.

This paper aims to build an application that extracts the grammatical rules needed to build the component responsible for transferring the surface syntactic structure into deep structure and vice versa; the so-called TT module referred to earlier. The examples used in this paper are derived from the English Penn tree-bank, however, this does not imply that the process or the application are only applicable to English; English was mainly chosen for the purpose of illustration and clarity. Section 2 discusses the design and implementation of application and the design of the data selected for the application. Section 3 discusses how the output of this application fits within the other modules in the UNLdev environment discusses the challenges. Section 4 evaluates the results of the generated and analyzed sentences. And finally section 5 is a conclusion and a survey of the future work.

2 AUTOMATIC UNL GRAMMAR EXTRACTOR

An application was built to automatically extract TT rules for both the analysis and generation processes. Its main objective is to allow UNL grammar developers to establish a more empirical and authentic grammar by means of massive analyzed data that cover the main syntactic structures of a language.

2.1 CORPUS COMPILATION

The Treebank or the parsed corpus is a text annotated with syntactic structure commonly represented as a TREE STRUCTURE. Treebanks can be ANNOTATED manually or semi-automatically. Examples of the available treebanks are the BULTREEBANK, the Penn Treebank and the QURANIC ARABIC DEPENDENCY TREEBANK…….etc.

The corpus used in the development of our application is derived from the English Penn Treebank. The English Penn Treebank annotates phrase structures[2]; for example, the syntactic analysis for ‘peter loves Mary’ following the PENN TREEBANK notation, is shown in figure 1 and may be represented by labeled brackets as shown in figure 2.

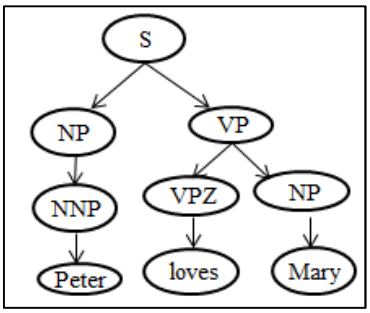

Figure (1) Penn treebank analysis for "peter loves Mary"

```
(S (NP-(NNP Peter))
   (VP(VPZ    loves)
      (NP -   NNP Peter))))
```

Figure (2): Penn treebank analysis for "peter loves Mary"

120 structures have been chosen from the Penn Treebank to constitute the data that will be used in the development of our application. These structures are divided into 113 full sentences and 10 phrases. The phrases include noun phrases (NPs), verb phrases (VPs) and adverbial phrase (APs). The following are five types of noun phrases;

[1] A bank of texts that are annotated syntactically and are commonly represented as tree structures; hence the name Treebank.
[2] more information about the Penn Treebank can be found at http://www.cis.upenn.edu/~treebank/

1. Proper nouns
2. Pronouns
3. Det + noun
4. Det + adj + noun
5. Det + adj + adj + noun

VPs include verbs in the past tense, Verbs in present tense and auxiliaries + verb. While APs include adverbs only. The structures of the 113 sentences are mainly generated from the rules (1), (2), (3) and (4):

(1) S → NP + VP
(2) NP → DET + (Adj) + Noun
(3) VP → V + (NP) + (AP)
(4) AP → Adverb

Out of these rules, it is possible to generate 113 sentences since the structures between brackets are optional. It is possible to generate from these rules a numerous amount of structures such as in table1:

Prpoernoun + aux + verb	Det + adj + noun + aux + verb +Pronoun + ADV
Pronoun + aux + verb	Det + adj + noun + aux + verb + Det + noun + ADV
Det + noun + aux + verb	Det + adj + noun + aux + verb +Det + adj + noun + ADV
Det + adj + noun + aux + verb	Det + adj + noun + aux + verb Det + adj + adj + noun + ADV
Det + adj + adj + noun +aux + verb	Det + adj + adj + noun + aux + verb +Pronoun
Prpoernoun + aux + verb + adv	Det + adj + adj + noun + aux + verb + Det + noun
Pronoun + aux + verb + adv	Det + adj + adj + noun + aux + verb +Det + adj + noun
Det + noun + aux + verb + adv	Det + adj + adj + noun + aux + verb Det + adj + adj + noun
Det + adj + noun + aux + verb + adv	Det + adj + adj + noun + aux + verb +Pronoun + ADV
Det + adj + adj + noun +aux + verb + adv	Det + adj + adj + noun + aux + verb + Det + noun + ADV
Prpoernoun + aux + verb +Pronoun	Det + adj + adj + noun + aux + verb +Det + adj + noun + ADV
Prpoernoun + aux + verb + Det + noun	Det + adj + adj + noun + aux + verb Det + adj + adj + noun +
Prpoernoun + aux + verb +Det + adj + noun	ADV
Prpoernoun + aux + verb Det + adj + adj + noun	Prpoernoun + verb
Prpoernoun + aux + verb +Pronoun + adv	Pronoun + verb
Prpoernoun + aux + verb + Det + noun + adv	Det + noun + verb
Prpoernoun + aux + verb +Det + adj + noun + adv	Det + adj + noun + verb
Prpoernoun + aux + verb Det + adj + adj + noun + adv	Det + adj + adj + noun + verb
Pronoun + aux + verb +Pronoun	Prpoernoun + verb
Pronoun + aux + verb + Det + noun	Pronoun + verb + adverb
Pronoun + aux + verb +Det + adj + noun	Det + noun + verb +adverb
Pronoun + aux + verb Det + adj + adj + noun	Det + adj + noun + verb +adverb
Pronoun + aux + verb +Pronoun + adv	Det + adj + adj + noun + verb +adverb
Pronoun + aux + verb + Det + noun + adv	Prpoernoun + aux + verb +adverb
Pronoun + aux + verb +Det + adj + noun + adv	Pronoun + aux + verb + adverb
Pronoun + aux + verb Det + adj + adj + noun + adv	Det + noun + aux + verb +adverb
Det + noun + aux + verb +Pronoun	Prpoernoun + verb + Prpoernoun
Det + noun + aux + verb + Det + noun	Prpoernoun + verb + Pronoun
Det + noun + aux + verb +Det + adj + noun	Prpoernoun + verb + Det + noun
Det + noun + aux + verb Det + adj + adj + noun	Prpoernoun + verb + Det + adj + noun
Det + noun + aux + verb +Pronoun + adv	Prpoernoun + verb + Det + adj + adj + noun
Det + noun + aux + verb + Det + noun + adv	Prpoernoun + verb + Prpoernoun + adv
Det + noun + aux + verb +Det + adj + noun + adv	Prpoernoun + verb + Pronoun+ adv
Det + noun + aux + verb Det + adj + adj + noun + adv	Prpoernoun + verb + Det + noun+ adv
Det + adj + noun + aux + verb +Pronoun	Prpoernoun + verb + Det + adj + noun + adv
Det + adj + noun + aux + verb + Det + noun	Prpoernoun + verb + Det + adj + adj + noun + adv
Det + adj + noun + aux + verb +Det + adj + noun	Pronoun + verb + Prpoernoun
Det + adj + noun + aux + verb Det + adj + adj + noun	Pronoun + verb + Pronoun
Det + adj + noun + verb + Det + adj + adj + noun + adv	Pronoun + verb + Det + noun
Det + adj + adj + noun + verb + Prpoernoun	Pronoun + verb + Det + adj + noun
Det + adj + adj + noun + verb + Pronoun	Pronoun + verb + Det + adj + adj + noun
Det + adj + adj + noun + verb + Det + noun	Pronoun + verb + Prpoernoun + adv
Det + adj + adj + noun + verb + Det + adj + noun	Pronoun + verb + Pronoun+ adv
Det + adj + adj + noun + verb + Det + adj + adj + noun	Pronoun + verb + Det + noun+ adv

Det + adj + adj + noun + verb + Prpoernoun + adv Det + adj + adj + noun + verb + Pronoun+ adv Det + adj + adj + noun + verb + Det + noun+ adv Det + adj + adj + noun + verb + Det + adj + noun + adv Det + adj + adj + noun + verb + Det + adj + adj + noun + adv Det + adj + noun + verb + Prpoernoun Det + adj + noun + verb + Pronoun Det + adj + noun + verb + Det + noun Det + adj + noun + verb + Det + adj + noun Det + adj + noun + verb + Det + adj + adj + noun Det + adj + noun + verb + Prpoernoun + adv Det + adj + noun + verb + Pronoun+ adv Det + adj + noun + verb + Det + noun+ adv Det + adj + noun + verb + Det + adj + noun + adv	Pronoun + verb + Det + adj + noun + adv Pronoun + verb + Det + adj + adj + noun + adv Det + noun + verb + Prpoernoun Det + noun + verb + Pronoun Det + noun + verb + Det + noun Det + noun + verb + Det + adj + noun Det + noun + verb + Det + adj + adj + noun Det + noun + verb + Prpoernoun + adv Det + noun + verb + Pronoun+ adv Det + noun + verb + Det + noun+ adv Det + noun + verb + Det + adj + noun + adv Det + noun + verb + Det + adj + adj + noun + adv

Table (1): The structures can be generated from rules (1), (2), (3) and (4)

2.2 DEEP-STRUCTURE RULE EXTRACTOR

The main objective of this application is to allow UNL grammar developers to establish a more empirical and authentic grammar. This can be achieved by means of massive analyzed data that cover the syntactic structures of a language. The application is designed to be flexible and interactive for the grammar developer. Firstly, the user should determine the kind of structure to be handled; whether a full sentence or a certain phrase. The user has to select whether the required rules are for the generation process or the analysis process, since TT rules are not the same in both processes.

After selecting the structure to be dealt with, the application reaches the processing phase which is composed of two levels; the phrase level and the sentence level; according to the structure of the input, the engine will decide which level is suitable. For each level there is a set of conditions and decisions that will be discussed later in sub section 2.3. Figure (3) is a simple diagram that illustrates how the input is processed in both levels.

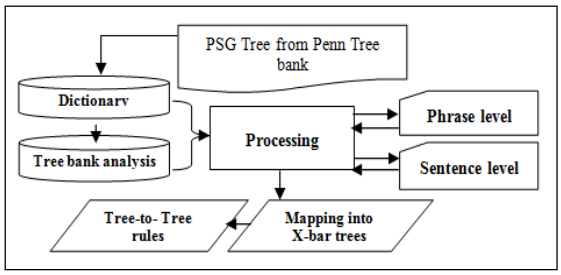

Figure (3): a simple diagram for algorith of the application

After the grammar developer selects the structure to be processed, the next step is for the engine to determine the corresponding phrase structure; in other words, the application would search its memory for the PSG structure most appropriate to this sentence according to Penn Treebank analysis. This task would be done with the help of a dictionary that contains words and their part of speech. The processing phase itself would be accomplished according to the type of the structure; whether the input is a phrase or a full sentence. As mentioned before, the main role of tree-to-tree rules is transforming the deep syntactic structure into a surface syntactic structure according to the principles of X-bar theory as the adopted theory in the UNL framework. Thus, there must be an intermediate phase before extracting TT rules from the annotated sentence since the annotated sentences are parsed according to Phrase Structure Grammar Structure PSG grammar. This intermediate process will map the annotated input of the Penn Treebank from PSG analysis to X-bar. This mapping is a challenging task since it is not one-to-one mapping process. The challenges of this phase will be discussed latter in sub section 2.3.

Our application depends mainly on a set of situations and decisions; in other words, it makes a specific decision to convert a branch of the PSG analysis into X-bar analysis according to the set of conditions present. As mentioned before, mapping from PSG to X-bar is not one-to-one; you cannot simply convert the branch named NP in PSG into a branch named NB in X-bar.

To demonstrate this, we are going to take the sentence "she kicked the ball strongly" as an example. The previous flowchart in figure (3) will explain the steps followed in mapping process. Firstly, the PSG input for this sentence would be as shown in figure (4):

(S (NP-SBJ She) (VPkicked (NP the ball)(ADVP-MNR strongly)))

Figure (4): The Penn treebank analysis for "she kicked the ball strongly"

Now we have four situations and four decisions accordingly as illustrated in figures (5), (6), (7), (8). The first decision has to do with the branch named "(S (NP... (VP... ", as shown in figure (4). If a sentence consists of a noun phrase and a verb phrase, then, it begins with (S(NP... (VP... in PSG analysis which should be converted initially to "?P(NP" and "VP" as shown in figure (5) for decision 1. The type of the phrase "?P" will not be defined here since it will defined later when more information is available.

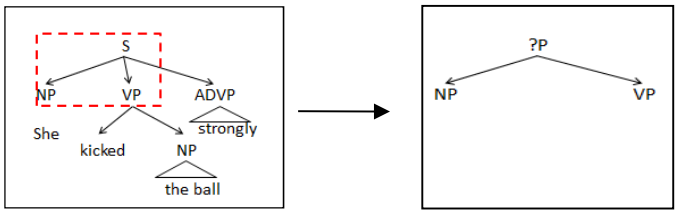

Figure (5): situation 1 and decision 1

The second situation faced here is the existence of the verb phrase and adverb phrase in "(VP … (ADVP…". As shown in PSG analysis in figure (6), the VP and the ADVP are sisters from the same mother node S. However, in order to convert it to X-bar, the decision was to make the ADVP an adjunct that would be a sister node to the V bar not the VP.

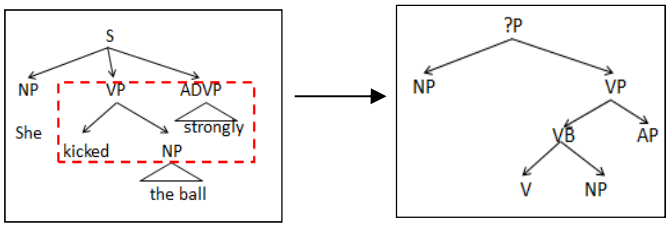

Figure (6): situation 2 and decision 2

The third situation has to do with the inflectional morpheme attached to the verb "kick". According to the dictionary, the verb "kick" is a regular verb in the past tense. This would define the "?P" as being inflectional phrase "IP" as shown in figure (7).

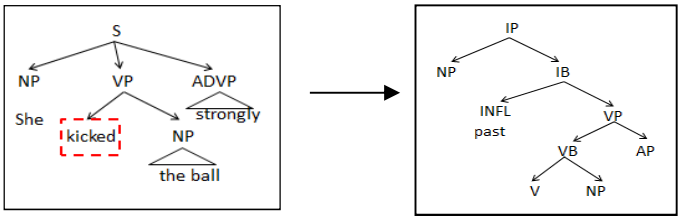

Figure (7) situation 3 and decision 3

Finally, the last decision will be devoted to place the terminal nodes "she – kick - the - ball - strongly" in their slots as shown in figure (8).

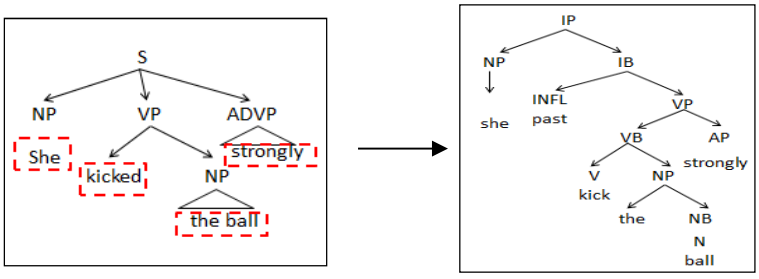

Figure (8) situation 4 and decision 4

Hence, the final X-bar tree corresponding to the PSG tree "(S (NP-SBJ She) (VPkicked (NP the ball)(ADVP-MNR strongly)))" would be represented as: "IP (NP-she IB(INFL-past VP(VB(V-kick ;NP(DET the NB(N-ball)))AP-strongly)))", and from this representation the TT rules would be extracted for both analysis or generation.

2.3 REMARKS ON MAPPING FROM PHRASE STRUCTURE GRAMMAR TO X-BAR

The X-bar theory is discussed in almost all modern textbooks of syntax, and it is assumed as a theory of Phrase Structure Grammar. PSG does not have intermediate categories which are larger than a word but smaller than a phrase. Such a category is needed because there are such units that could not be classified neither as a phrase nor as a category. But Phrase Structural Grammar treats such unit in the same manner it treats a category[3]. The X-bar abstract schema is illustrated in figure (9):

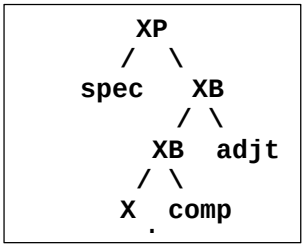

Figure (9): the schema of X bar theory

In figure (9), X is the head of the whole syntactic structure; it is derived (or projected) out of it. The letter X is used to as an arbitrary lexical category. When analyzing a specific utterance, specific categories are assigned, and the X may become an N for noun, a V for verb, an J for adjective, or a P for preposition. The "comp" stands for complement and it is an internal argument which is necessary to the head to complete its meaning. The "adjt" stands for adjunct; it is a word, phrase or clause which modifies the head but which is not syntactically required by it. Finally, the "spec" stands for specifier and it is an external argument which qualifies (determines) the head. XB (X-bar) is the general name for any of the intermediate projections derived from X. XP (X-bar-bar, X-double-bar, X-phrase) is the maximal projection of X.

As mentioned before, X-bar theory is assumed to be a theory of PSG, however, mapping from PSG to X-bar analysis is not a direct endeavor. In contrast with X-bar syntax, Phrase Structure rules cannot distinguish between elements that are subcategorized by the head and those that are optional. For example, in the sentence "A student of physics with a long hair", it is not possible to determine if the prepositional phrase "of physics" is a complement of the head "student" or an adjunct that is optional. However, in X-bar theory, the complement is a sister of the head, and the constituent formed by a head and its complement are labeled as units named XB (X-bar). As shown in figure (10), in PSG analysis, the two prepositional phrases both originate from the same node NP; on the other hand, in figure (11), according to X-bar analysis, the prepositional phrase "of physics" is a sister node of the head and is its complement, and the prepositional phrase "with long hair" originates from the node NP since it is an optional adjunct.

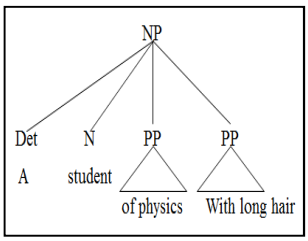

Figure (10) representation of the structure using PSG

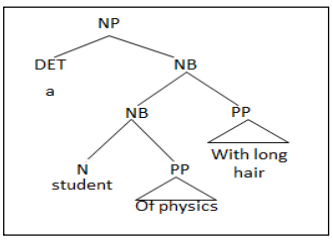

Figure (11) representation of the structure using X-bar

Another note on PSG analysis is regarding the specifiers. Specifiers in PSG are still part of the constituent XP. X-bar specifiers are also XPs; however, the specifier is daughter to the XP and sister to the XB. Referring back to figures (10) and figure (11), the specifier "a" in PSG analysis is a daughter of the NP but is a sister to N, PP and PP. In X-bar analysis, the specifier is a daughter of NP and a sister to NB.

[3] http://epistemic-forms.com/ps-xbar.html

The final note regarding X-bar theory is the adjuncts. Adjuncts can be sisters to XB and daughters to XB. PSG lacks the concept of distinguishing adjuncts. As shown in figures 4 and 5, the prepositional phrase "with long hair" is well defined in X-bar analysis since it is the sister of the NB and the daughter of another NB. In contrast to PSG, the preposition phrase originates directly from the NP.

All the previous observations make the task of mapping a whole analysis from PSG to X-bar quite challenging. The grammar developer should be fully aware of the concepts of specifiers, complements and adjuncts to the extent that he/she is able to determine whether a certain unit is a complement or an adjunct even if two units are from the same category "PP" and originate from the same mother "NP".

3 TESTING THE APPLICATION

The main goal of this application is to provide a set of automatically generated rules that would constitute the Tree-to-Tree module which is the most important module in the UNL grammar. This experiment was conducted on English as the source language. The sample of sentences was selected from the English Penn Tree Bank version II [4] as mentioned in sub section 2.1. The collected sample was chosen to represent examples of phrases and complete sentences. In order to test the validity of this application, the generated rules (both for generation and analysis process) were supposed to be input into the UNLdev[5]. We must indicate here that special care have been taken to make sure the automatically extracted rules are in the same format as those already within the UNLdev to ensure homogeneity. The UNLdev is a web-based integrated development environment for creating and editing dictionary entries and grammar rules in order to be used in natural language processing applications. The generation rules were added to EUGENE; the engine responsible for producing natural language texts out of UNL documents, while analysis rules were added to IAN; the engine responsible for producing that produces UNL documents out of natural language texts. The generated rules are imported into the UNLdev environment and saved in a separate text file in UTF8 encoding. The user can then upload the imported rules into EUGENE if they are generation rules, or IAN if they were analysis rules.

We will now show the testing of a sample of the generation TT rules that were extracted automatically and uploaded into EUGENE. Figure (12) shows a screen shoot of EUGENE where a group of rules named "Testing Generation" was created. Within this group there are five rule-sets; NN, NT, TT, TL and LL. All the rules-set are ready except the TT rule-set, it is still empty so that the grammar developer can add the automatically extracted TT rules to it. Figure (13) shows the same for IAN, where TT rule-set is still empty so that the analysis TT rules would be added to it.

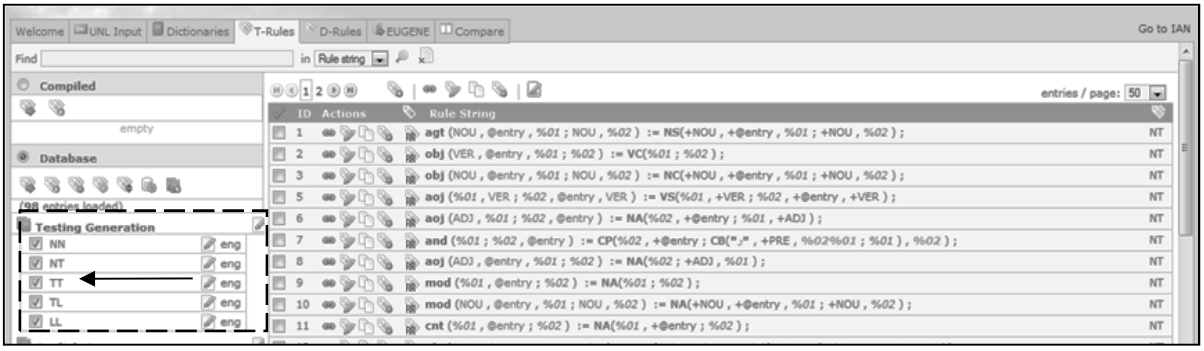

Figure (12) A sample of the grammar in the Eugene engine

4Bracketing Guidelines for Treebank II Style, Penn Treebank Project.

5 You can reach it at http://www.unlweb.net/wiki/UNLdev

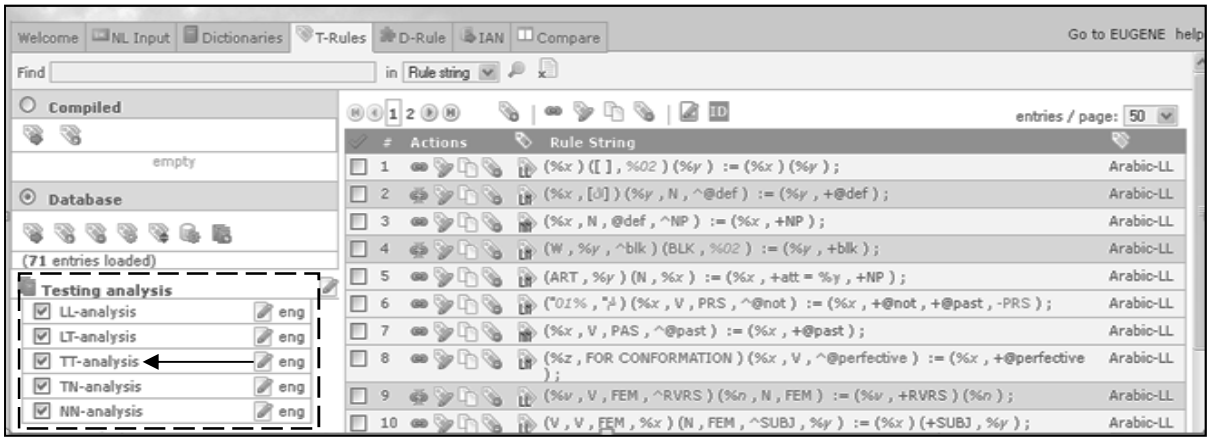

Figure (13): A sample of the grammar in the IAN engine

After preparing these rules-sets in addition to the proper dictionary and the input file to be processed, it is time to test the validity of the automatically generated TT rules. Figure (14) shows the UNL graph that expressing the sentence: "He killed her with a knife in the kitchen yesterday" as an example. In EUGENE, the task is to convert the UNL graph into a natural language string while IAN is supposed to generate the UNL graph for the sentence.

```
[S:8]
{org}
he killed her with a knife in the kitchen yesterday.
{/org}
{unl}
obj(201323958:03.@past, 00:05.@3.@female)
ins(201323958:03.@past, 103623556:0B.@indef)
plc(201323958:03.@past, 103619890:0H.@def)
tim(201323958:03.@past, 115156187:0J)
agt(201323958:03.@past, 00:01.@3.@male)
{/unl}
[/S]
```

Figure (14): The UNL graph of the sentence "He killed her with a knife in the kitchen yesterday"

The NN and NT rules convert the semantic graph in figure (14) into five syntactic branches that are represented in the graph in figure (15):

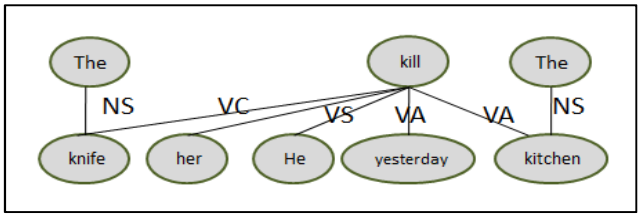

Figure (15) the counterpart syntactic relations of the semantic relations in figure (14)

The next step is to add the TT rules that would organize these branches into the surface syntactic structure of the sentence according to the X-bar principles. Finally, the TL and LL rules would list this tree into a natural language sentence. For the analysis process, rules will be added to IAN and the steps will be reversed. In the following section, we will test the output of the automatically extracted TT module when being combined with four other manually built modules and using the whole grammar (composed of five modules altogether) to process a number of sentences and save its output to be evaluated later by comparing to the output of another grammar in which all five modules were built manually.

4 EVALUATION

In the UNL System, the F-measure (or F1-score) is the measure of a grammar's accuracy. F-measure is measured by means of considering both the precision and the recall of the grammar to compute the score, according to the formula in figure (16):

F measure = 2 x ((precision x recall) / (precision + recall))

Figure (16) the formula that calculate the F-measure

Precision is the number of correct results divided by the number of all returned results while recall is the number of correct results divided by the number of results that should have been returned. The f-measure can be calculated automatically in the UNLarium[6]. It is possible to compute the F-measure both for analysis where the output is a UNL graph, and in generation where the output is a list of natural language words as shown in figure (17).

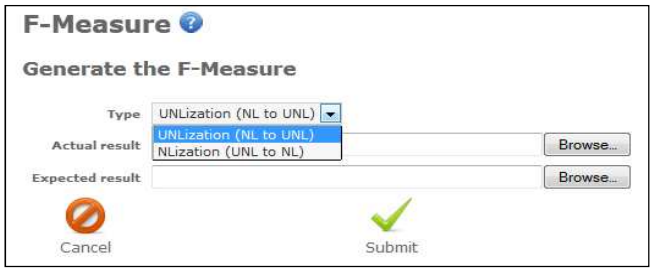

Figure (17) A screenshot for F-measure

The result is considered correct when the Levensthein distance between the actual result to be expected and the expected result is less than 30% of the length of the expected result. The Levensthein distance is defined as the minimal number of characters you have to replace, insert or delete to transform a string (the actual output) into another one (the expected output).

For this process, two documents are required. The documents must be in plain text format (.txt) with UTF-8 encoding. One document is for the actual result extracted automatically by our application while the other is for the result obtained from the manually built grammar in the UNLdev. The actual and expected results must be UNL documents in case of analysis, or documents in the same natural language in case of generation and both must not have been post-edited. The table in figure (18) illustrates the F-measure resulting from comparing the output generated from the automatically extracted grammar with that of the manually built grammar.

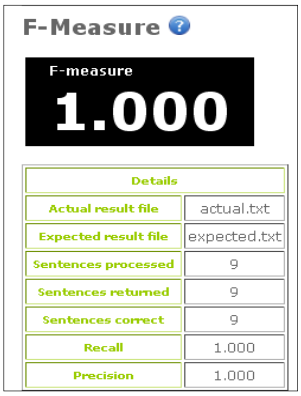

Figure (18) The F-measure comparison result

5 CONCLUSION AND FUTURE WORK

Tree-to-Tree rules are the most complex in UNL grammar and represent the core of the transformation rules in UNL either in generation or analysis. This paper presents a corpus-based application which automatically extracts tree-to-tree rules from syntactically annotated sentences. The process of extraction goes through an intermediate state where the annotated sentences are converted form PSG analysis into X-bar. After the conversion into X-bar, it is possible to use the extracted rules in IAN or EUGENE; the analysis and generation engines in the UNLdev. In the future, the extracted rules can be used in the automatic updating of grammars.

[6]http://www.unlweb.net/unlarium/

REFERENCES

[1] Andr´asKornai, Stanford University. The x-bar theory of phrase structure. Hungarian Academy of Sciences, Geoffrey K. Pullum. University of California, Santa Cruz.
[2] Michelle Sheehan. From Phrase Structure Rules to X-Bar Theory. 18th February 2010
[3] Joybrato Mukherjee, " Corpus linguistics and English reference grammars", Justus Liebig University, Giessen
Philippe Blache, Marie-Laure Guénot& Tristan van Rullen. A corpus-based technique for grammar development. LPL-CNRS, Université de Provence.
[4] N. Chomsky. “Remarks on Nominalization” In: Readings in English Transformational Grammar. R. Jacobs and P. Rosenbaum (eds.), pp. 184-221. 1970.
[5] Noha Adly, Sameh Alansary, “Evaluation of Arabic Machine Translation System based on the Universal Networking Language”, in proceedings of 14th International Conference on Applications of Natural Language to Information Systems, (NLDB 2009), Saarland University, Saarbrücken-Germany, June 24 - 26 2009.
[6] R. Martins, and V._ Avetisyan, “Generative and Enumerative_ Lexicons in the UNL Framework”, in proceedings of 7th International Conference on Computer Science and_Information Technologies, (CSIT 2009), 28 September - 2 October, 2009, Yerevan, Armenia. 2009.
[7] http://www.unlweb.net/wiki/Grammar_Specs
[8] http://www.unlweb.net/wiki/Relations
[9] http://epistemic-forms.com/ps-xbar.html
[10] http://www.unlweb.net/wiki/UNLdev
[11] http://www.cis.upenn.edu/~treebank/