UNL Editor: An Annotation tool for Semantic Analysis

Sameh Alansary[*1], Magdy Nagi[**2], Noha Adly[**3]

[*]*Phonetic and Linguistics Department, Faculty of Arts, University of Alexandria*
ElShatby, Alexandria, Egypt

[*]*Bibliotheca Alexandrina, Alexandria, Egypt*

[1]sameh.alansary@bibalex.org

[**]*Computer and System Engineering Department, Faculty of Engineering*

Alexandria University, Egypt

[2]magdy.nagi@bibalex.org

[3]noha.adly@bibalex.org

Abstract— **This paper presents the UNL Editor as a tool for semantic annotation; discussing and describing the tool in details. The paper regards the tool in two aspects, describing its linguistic framework; explaining the logic on which the UNL Editor is based upon. Then, it goes to explain how this logic is applied when carrying out the semantic annotation of the natural language texts through presenting step by step instruction for using the tool. Finally, it exhibits the different usage of such a tool. However, in order to control the size of the paper, this paper is not concerned with addressing different linguistic issues of annotating natural language tests, or the linguistic difficulties arising within the process; it is only limited to presenting linguistic capabilities of the tool to prove its efficiency in semantic annotation[1].**

1 Introduction

In the recent years, semantic Annotation has become an increasingly important research topic being a fundamental element of many Natural Language Processing applications like information retrieval, query answering and information extraction. Semantic annotation is additional information in a document that identifies or defines the semantics of a part of that document. In other words, Semantic annotation is about attaching sense tags, names, attributes, comments, descriptions, etc. to a document or to a selected part in a text [1]. Consequently, helping to bridge the ambiguity of the natural language when expressing notions and their computational representation in a formal language; by telling a computer how data items are related and how these relations can be evaluated. Thus, opening the way to numerous applications.

With rapidly growing amount of on-line web documents, web users need to find, share, and combine information more easily; urging researchers to focus on the creation and dissemination of innovative Semantic Web technologies to facilitate automated processing [2]. The semantic web depends entirely on semantic annotation. Hence, it would only seem natural to find number of tools designed to perform full semantic annotation for natural language texts. However, this is not the case, the number of tools intended to perform semantic annotation is extremely limited [3]. There have been several attempts to create a tool for analyzing natural language texts semantically. Some of the most worth of noting applications are GATE [4], KIM [5], Melita [6]. Nevertheless, none of the tools are totally automatic. Furthermore, these systems perform annotation on words and terminologies to indentify real world objects and their relationship in the text. None of them provide annotation above word level. A brief overview of some of these tools:

GATE (General Architecture for Text Engineering) is an infrastructure for development software components based on Human Languages. The GATE system provide many functionalities among them, it provides the functionality to annotate textual documents both manually and automatically. GATE uses JAPE [7] pattern matching engine for rule based Named Entity Recognition. JAPE is ontologically aware which can map the Named Entity to ontology classes during recognition. In GATE, the task of textual annotation is just defined more domain specific rules in addition to already available basic rules.
KIM is another ontology base semantic annotation system that uses a special knowledge base (KIMO) which has been pre-populated with 200,000 entities. KIM uses GATE, SESAME and Lucene for many information extraction tasks. KIM also uses version of ANNIE for Named Entity Recognition. KIM has a feature of automatically adding new instances found in text to Ontology. It also performs disambiguation step because many instances can be added to different places in ontology.

[1] Different issues of annotating Arabic texts semantically will be found in: [1] S. Alansary, "Semantic Annotation of Arabic Texts: Issues and Implications" (forthcoming)

Melita provides the interface to semantically annotate the textual document using Adaptive Information Extraction technique. This technique reduces the burden of text annotation on user. It starts with manual annotation of text by user and as user keeps on annotating text the system learns the annotation process. Melita uses Amilcare [8] which runs in background learning how to reproduce the inserted annotation.
Considering that semantic annotation has became a comprehensive concept, number of attempts have been made in order to integrate linguistic approaches in the analysis of natural language corpus, some of the most representative results were the Propbank project [9], FrameNet project [10]. The Proposition Bank project (Propbank) focuses on the argument structure of verbs and adding a layer of predicate-argument information, or semantic role labels, to the syntactic structures of the Penn Treebank. It aims to provide a broad-coverage hand annotated corpus with semantic annotation, enabling the development of better domain-independent language understanding systems. The FrameNet was initially a lexicographic project, engaged in building a lexicon with uniquely detailed information on the syntax and semantics of Lexical Units. More recently, since 2004 FrameNet has also been annotating continuous texts for deep semantic annotation. The FrameNet approach is based on linguistics theory of frame semantics. However worthy these attempts were, they were all manually done; none of which was performed by tools. Thus, the need to provide a tool designed with the intention of performing semantic analysis became undeniably clear.

In the context of the UNL (The Universal Networking Language), a semantically based interlingua to break language barriers between human languages, the UNDL Foundation in co-operation with Bibliotheca Alexandrina has started an initiative for building a tool for semantic annotation called the UNL Editor; a visual editor designed with the intention of providing full semantic annotation, thus analyzing natural language texts and, generating UNL documents. This tool is based upon a comprehensive visualization of the entire process of the annotation. It is uniquely designed on linguistic background; adopting certain linguistic theories closely related to computational linguistics in terms of using unified super sets of semantic relations [11] thus overcoming the problem of conflicting and confusing names [12], and making use of renowned lexical recourses; WordNet [13]. Moreover, it provides a powerful visual interface for working with UNL data both in a textual and graphical mode with friendly interface creating an appropriate environment for navigating through the needed steps of providing the analysis; it offers a visualization of the analysis through graphs which aids the representation of the semantic network created with every sentence analyzed. Most importantly, the UNL Editor`s output offers the much need training data for semantic annotation due to the fact that the relations and concepts used are clearly defined as well as standardized within the UNL Editor framework, in addition the output is presented in a text file that could be easily used. The UNL Editor exhibits enormous flexibility and opportunities in handling natural language text due to the fact that it is designed upon linguistic framework, minding the complexity and richness of natural language, thus enriching the tool with all different kinds of options in order to handle the natural language, and paving the way for other applications through its easy to be used output.

This paper is concerned with presenting and explaining the UNL Editor as a manual tool for semantic annotation. It is divided into four sections; section 2 exhibits the linguistic framework which the design of the UNL Editor adopts as its bases; indicating why it is designed as such and linguistic theories are been adopted, section 3 is a detailed explanation accompanied with screenshots illustrating how this application could be used, section 4 represents the different usages of the UNL Editor as a tool for semantic annotation. Finally, Section 5 concludes the paper.

2 Linguistic Framework

The UNL Editor provides a means enabling the analysis of the underlying semantic relations composing the Natural Language sentences. It is designed on linguistic bases . On a semantic assumption or rather on semantic theory stating that a deep semantic analysis for a natural language text requires two levels of semantics; lexical semantics and grammatical semantics [14].

A. Lexical Semantics

It is the study of how and what the words of a language denote. In other words, lexical semantics is meaning at word level [15]. In the UNL Editor, lexical semantics is expressed through creating the nodes, a process in which every word or rather every concept in the sentence to be analyzed is matched with its corresponding ID, meaning that a single node may contain more than one lexical item; a compound word, as long as it is representing a single concept. For example the term "Holy Quran" represents single concept, therefore it would be considered one node, having a single ID. The ID is a nine-digit string that is distinct number and assigned to each concept. The dictionary, from which the IDs are extracted, is based upon the WordNet 3.0; a lexical database for English Language, contains 155,287 words organized in 117,659 synsets for a total of 206,941 word-sense pairs). The WordNet is considered to be the most prominent and widely used lexical resource for researchers in computational linguistics, text analysis, and many related areas [16]. In order to make the process of selecting the appropriate ID easier and for

more clarification to the concept, the UNL Framework made use of the set of information the WordNet attach to each concept, these information consist of a distinct ID, an abstract meaning (the gloss), the "synset" which is a set of one or more synonyms that are interchangeable in some context without changing the truth value of the proposition in which they are embedded, the corresponding part of speech and in some cases examples are shown. The right half of the interface is dedicated for the lexical semantics through the search pane, in which there are three search options are offered by exhibiting three tabs, each tab is dedicated for a different kind of search [3.1]. One of which offers the possibility of uploading dictionaries in attempt of providing an integrated development environment for UNL.

B. Grammatical Semantics

It has to do with meaning at sentence level; grammatical semantics is the study which explores the relation between patterns of meaning and grammatical structure. It is based on the assumption that the syntactic structure of the sentences overlaps with its semantics [17]. In the UNL Editor, grammatical semantics is expressed in terms of a range of semantic relations, and a list of attributes. There has always been a problem with using semantic relations as there is no formal basis for defining the notion clearly, making determining what should be qualified as a semantic relation and what is not confusing. In order to overcome this problem, the UNL Editor has proposed a unified super set of the semantic relations. These relations are highly standardized as each relation is clearly defined in the UNL framework. Table 1contains all the 45 semantic relation that the tool includes and they are a closed set of relations. Moreover, it is a directed graph meaning that every relation has to start from certain node in order to convey the correct meaning. Relations are used to describe the objectivity information of sentences. In the UNL, relations are normally regarded as representations of semantic cases or thematic roles (such as agent, object, instrument, etc.) between concepts. They are used in form of arcs connecting a node to another node in a UNL graph. They correspond to two-place semantic predicates holding between two concepts. Relations are represented as two or three-character lower-case strings. Since there are similarities between the semantic relations and syntactic relations in name and function, it may seem that the labels used for relations are different names for special grammatical functions. However, the intention is that the labels used denote specific ideas rather than grammatical structures, the conceptual relations used in UNL are much more abstract than the grammatical relations found in syntax. In general, relations are always used to describe semantic dependencies between syntactic constituents. For example, in a sentence like “John breaks the door”, the syntactic subject of the sentence is “John” and semantically it would be regarded as the “agt”, whereas in a sentence like “the sugar melts in tea” the lexical item “sugar” is the syntactic subject of the sentence but semantically it would be considered as an object “obj”.

Table I illustrates the UNL Editor semantic relations; definition, description and example to each relation

TABLE I
SEMANTIC RELATIONS

RELATION	DEFINITION	DESCRIPTION	EXAMPLE
Agt	Agent	a thing which initiates an action	car runs
And	And	a conjunctive relation between concepts	John and Mary
Aoj	thing with attribute	a thing which is in a state or has an attribute	Leaf is red
Bas	Basis	a thing used as the basis(standard) for expressing degree	Ten is three more than seven
Ben	Beneficiary	a not directly related beneficiary or victim of an event or state	To give one’s life for one’s
Cag	co-agent	a thing not in focus which initiates an implicit event which is done in parallel	To walk with John
Cao	co-thing with attribute	a thing not in focus is in a state in parallel	be with you
Cau	Cause	the cause of a state	The cause of the accident….
Cnt	Content	an equivalent concept	The Internet: an amalgamation
Cob	affected co-thing	a thing which is directly affected by an implicit event done in parallel or an implicit state in parallel	dead with Mary
Con	Condition	a non-focused event or state which conditioned a focused event or state	if you are tired, we will go straight home
Coo	co-occurrence	a co-occurred event or state for a focused event or state	was crying while running
Dur	Duration	a period of time during an event occurs or a state exists	work nine hours (a day)

Equ	Synonym	Synonym	the deconverter (a language generator)
Fmt	Range	a range between two things	the alphabets from a to z
Frm	Origin	an origin of a thing	a visitor from Japan
Gol	goal/final state	the final state of object or the thing finally associated with object	the lights changed from green to red
Icl	Inclusion	Inclusion	a bird is a (kind of) animal
Ins	Instrument	the instrument to carry out an event	look at stars through a telescope
Int	Intersection	indicates all common instances to have with a partner concept	an intersection of tableware and cookware
Man	Manner	the way to carry out event or characteristics of a state	move quickly
Met	Method	a means to carry out an event	solve … with dynamics
Mod	Modification	a thing which restrict a focused thing	the whole story
Nam	Name	a name of a thing	his son "Hikari"
Obj	affected thing	a thing in focus which is directly affected by an event or state	the table moved
Opl	affected place	a place in focus where an event affects	pat … on shoulder
Or	Disjunction	disjunctive relation between two concepts	Will you stay or leave?
Per	proportion, rate or distribution	a basis or unit of proportion, rate or distribution	eight hours a day
Plc	Place	the place an event occurs or a state is true or a thing exists	cook … in the kitchen
Plf	initial place	the place an event begins or a state becomes true	traveling from Tokyo
Plt	final place	the place an event ends or a state becomes false	to travel to Boston
Pof	part-of	a concept of which a focused thing is a part	the preamble of a document
Pos	possessor	the possessor of a thing	John’s dog
Ptn	Partner	an indispensable non-focused initiator of an action	compete with John
Pur	purpose	the purpose or an objective of an agent of an event or a purpose of a thing which exist	come to see you
Qua	Quantity	a quantity of a thing or unit	Two cups of coffee
Rsn	Reason	a reason that an event or a state happens	They can start because Mary arrived
Scn	Scene	a virtual world where an event occurs or state is true or a thing exists	win a prize in a contest
Seq	Sequence	a prior event or state of a focused event or state	Look before you leap
Src	Source	the initial state of an object or thing initially associated with the object of an event	The lights changed from green to red
Tim	Time	the time an event occurs or a state is true	leave on Tuesday
Tmf	initial time	the time an event starts or a state becomes true	work from morning to [till] night
Tmt	final time	the time an event ends or a state becomes false	be full till tomorrow
To	Destination	a destination of a thing	a train for London
Via	intermediate place	an intermediate place or state of an event	go … via New York

Other additional information are being presented through attributes, representing information conveyed by natural language grammatical categories (such as tense, mood, aspect, number, etc) [18]. In opposition to relations, attributes correspond to one-place predicates; attributes are intended to be used as annotations made to nodes or hypernodes of a UNL hypergraph. Moreover, they are also a closed set. The names of attributes are always expressed in lower case words or expressions. Attributes are also used to express the range of concepts such as the concept indicate generic type of concept and so forth. One the one hand, relations and concepts are used to describe the objectivity information of sentences. On the other hand, attributes modify concepts or semantic networks to indicate subjectivity information such as about how the speaker views these states-of-affairs and his attitudes toward them and to indicate the property of the concepts. This includes phenomena technically called "speech acts", "propositional attitudes", "truth values", etc. They are used to express logical expressions in order to strengthen the expressibility of the UNL. Attributes are divided into the following groups:

1) Aspect	8) manner	15) register
2) Degree	9) modality	16) reference
3) document structure	10) numerals	17) social deixis
4) emotions	11) person	18) specfication
5) figure of speech	12) place	19) tense
6) gender	13) polarity	20) time
7) lexical category	14) quantification	21) voice

Attributes are mainly used to convey three different kinds of information. First, the information on the role of the node in the UNL graph, as in the case of '@entry', that indicates the main (starting) node of a UNL directed graph; secondly, The information conveyed by bound morphemes and closed classes, such as affixes (gender, number, tense, aspect, mood, voice, etc), determiners (articles and demonstratives), adpositions (prepositions, postpositions and circumpositions), conjunctions, auxiliary and quasi-auxiliary verbs (auxiliaries, modals, coverbs, preverbs) and degree adverbs (specifiers); thirdly, The information of the (external) context of the utterance, i.e., non-verbal elements of communication, such as prosody, sentence and text structure, politeness, schemes, social deixis and speech acts.

3 How to Use the UNL Editor?

This section will present step by step instruction for using the UNL Editor tool to create the semantic graph representation of the sentences. In order to use the tool, the user will have to sign in the UNL web then access the UNL Editor via UNL dev application (The UNL Integrated Development Environment). Reference [19] shows the advantages of the UNL Editor being a web application:

- no installation and updating is required
- easy access through the internet
- data is stored remotely, requiring little or no disk space from the part of the user
- easier to get collaboration possibilities and make contributions

Figure 1 describes the steps for reaching the semantic graphic representation. Within the UNL Editor Frame work, the process of decision making is completely human: the user uploads the text to be analyzed; selects the corresponding IDs; relate nodes through creating semantic relations; and assigns attributes to nodes. The first step will be the text input and text segmentation followed by concepts selection to create the nodes and adding the appropriate attributes to each node then the final step in order to reach the semantic graph will be linking the created nodes by semantic relations [20].

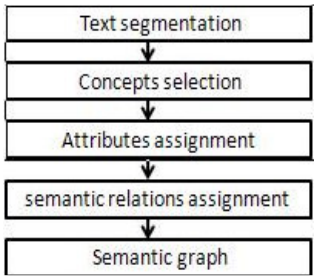

Figure 1: steps for reaching the semantic graph

A. Text Input

After accessing the UNL Editor, the first step is to add the natural language text that needs to be annotated this process could be achieved through two ways; either by selecting the option of "manual text input" in which the user will need to write or paste the source text into an editable area, or by selecting "upload a file" option to upload a file with either text contains UNL, the user wants to modify its content or, to upload plain text contents in order to be converted to the UNL. The contents of the file will be read and parsed into a UNL document format then these documents are presented as projects and are physically stored in the UNL Editor Data Base with the options of removing or downloading these projects, or of adding a new one. Finally, the document will be split into sentences, the UNL adopts some parameters such as "." for determining the end of the sentences and where the split should be. After the document is split into sentences, the sentences will be ready for the linguistic analysis. After the text has been uploaded and split into sentences, the interface will be divided into two parts; the left pane exhibits the previously saved documents in the upper part while the lower part contains the shared files between the application users, and the right pane contains the sentences that have been segmented. In the case of huge number of sentences which would be saved across many pages, the application provides the user with the ability to navigate between sentences by writing the sentence number in the navigation text box.

Segmented sentences can also be deleted by the "delete sentence button", the user can add a sentence in the document by the "add sentence button". Furthermore, the user could add any comment about the sentences in the comments text box. If there is a problem with the spelling or segmentation, as the application can split "e.g." since it considers "." as a delimiter and could segment after it, the user can modify the text by the "editing text button". Then, the sentences are ready to be annotated by using the UNL Editor; the user will have to use the "graph drawing button" to start annotation (see figure 2).

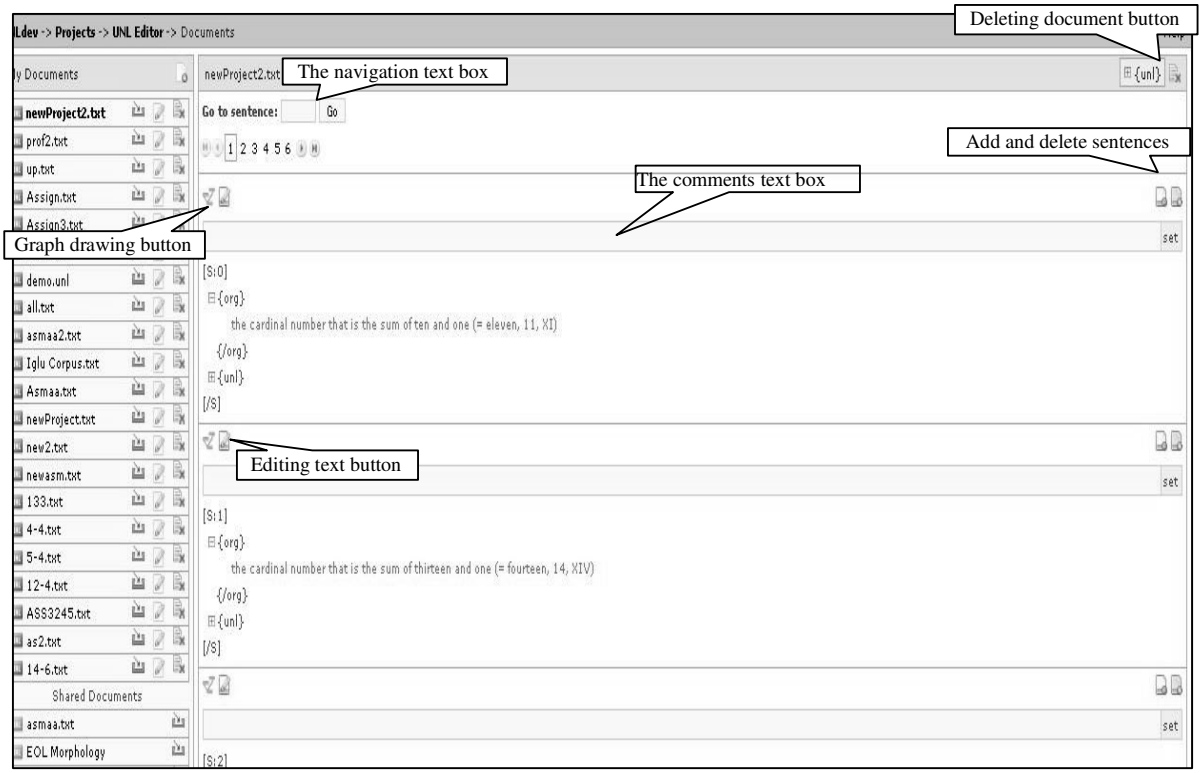

Figure 2: The UNL Editor interface presenting the segmented text

B. Nodes Creation

The first step for annotating a natural language text includes selecting the corresponding ID for each concept, and choosing the appropriate attributes the concepts need in order to complete the meaning of the concepts the sentence contains.
The first step for annotating a natural language text is determining which of the lexical items constituting the sentence represent concept and which do not; usually auxiliary verbs, model verbs and articles are not regarded as concepts and are being

represented by attributes, also the user should determine which constituents represent a compound word and which do not, for example "White House" it could mean the American presidential House or simply a house that is painted white, it is up to the user to decide according to the meaning. Furthermore, compound words may be separated by other units, for instance "look up" is a compound verb that could be separated as in "look the dictionary up". In this case, the user will have to determine the words that represent the intended sense and that should be included in a single node. Only after determining the concepts of the sentence, the user could create the nodes and choose the corresponding IDs.

The option of editing nodes is provided in the interface as after creating a node, the user may discover that this node is not needed in annotation, and needs to be deleted so the option of deleting nodes is provided through a button for deleting nodes "delete node button". Another button is provided for duplicating the nodes "clone node button" as some situation requires duplicating the same node as in the case of ellipsis; the omission from a sentence or other construction of one or more words that would complete or clarify the construction [21]. In a sentence as “I'm leaving and so does he” which means that “I'm leaving and he (is leaving) too” the node "leave" would have to be duplicated in order to represent the entire semantic graph of the sentence, and the attribute “@ellipsis” will have to be assigned to the node. Figure 3 illustrates the buttons needed in the process of creating the nodes.

1) Concept Selection: There are three possibilities for looking up the concept when working with the Graph Editor, provided through three tabs that enable the user to choose the method he believes the most appropriate. These three tab ranges from the most general to most specific dictionaries, the first tab is the concepts tab which enables the user to choose senses from the general dictionary uploaded from the WordNet, the second tab is the memory tab to choose from other previous users selections from the WordNet and the final tab is the dictionary tab in which the user uploads his own dictionary:

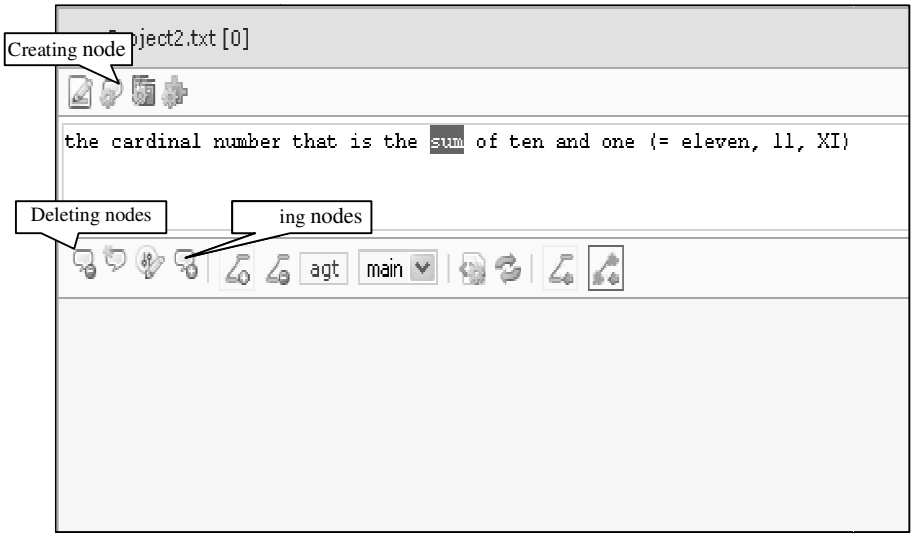

Figure 3: Creating, deleting and cloning the nodes

Concepts tab

This tab matches the lexical items included in the sentence with the concepts extracted from WordNet 3.0. In figure 4, the word "boy" is matched to all the different concepts that could be expressed with the lexical item "boy". In order to obtain a more precise idea about the matching concepts, more details are shown at pointing the mouse on each concept. A light preview appears containing; a distinct ID represented as a nine digit number, an abstract meaning (the gloss), a set of synonyms (the synset), the corresponding part of speech, the frequency and in some senses examples are shown. Moreover, the UNL Editor provides a filtering option in order to facilitate the process of searching; Users are able to search according to the part of speech either it is a noun, proper noun, verb, adjective, participle (A lexical item, derived from a verb, that has some of the characteristics and functions of both verbs and adjectives) or adverb, and for more flexibility there are three search options, the search could be performed according to specific word or string or number.

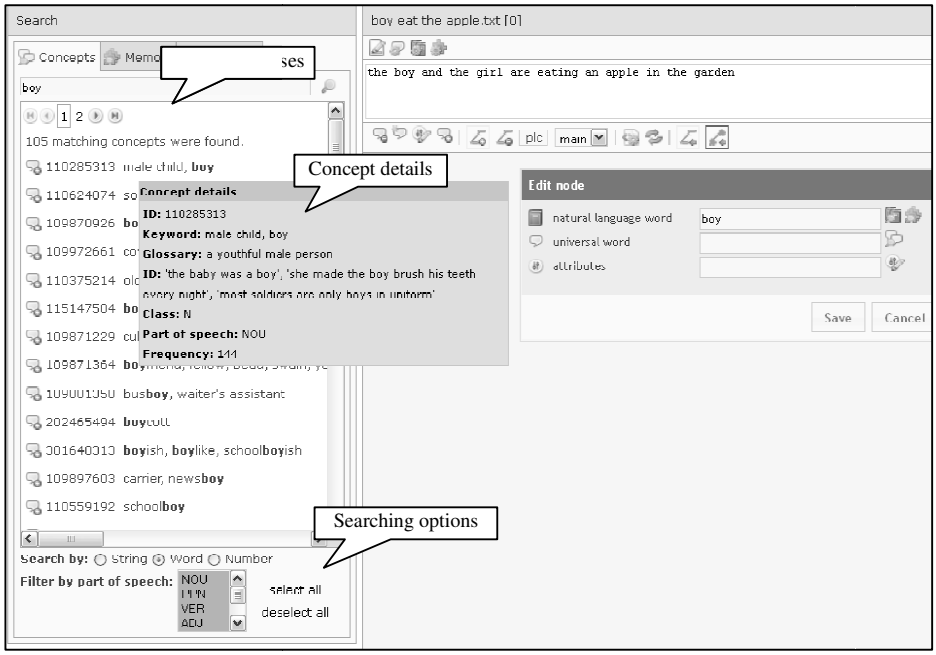

Figure 4: Searching for concept

If the user could not find the corresponding ID for the word, as in the case of names or websites or etc., the tool enables the user to handle this word as a temporary concept by putting that word between double quotes and it would be regarded as a node. The user may face other problem while creating the node as he may not find the appropriate sense for the lexical item; also he can add the node as a temporary concept, but to be added to the dictionary in the future.

Memory tab

This tab displays the dictionary Lookup memory that has the ability to store, retain, and recall nodes accumulated by all users who has used the UNL Editor as a tool to analyze natural language documents, the results show the matching concepts that were found. Unlike the results of the concept tab, the results displayed by the memory tab include the attributes that were assigned to the previously used IDs. The results of the memory tab are of a great use as it gives a clear idea about the frequency of usage of the different senses of the same lexical item, as well as it provides the user with a more feasible results since the concepts are accompanied with the needed attributes. Figure 5 shows the limited list of previously used senses of the concept “boy”.

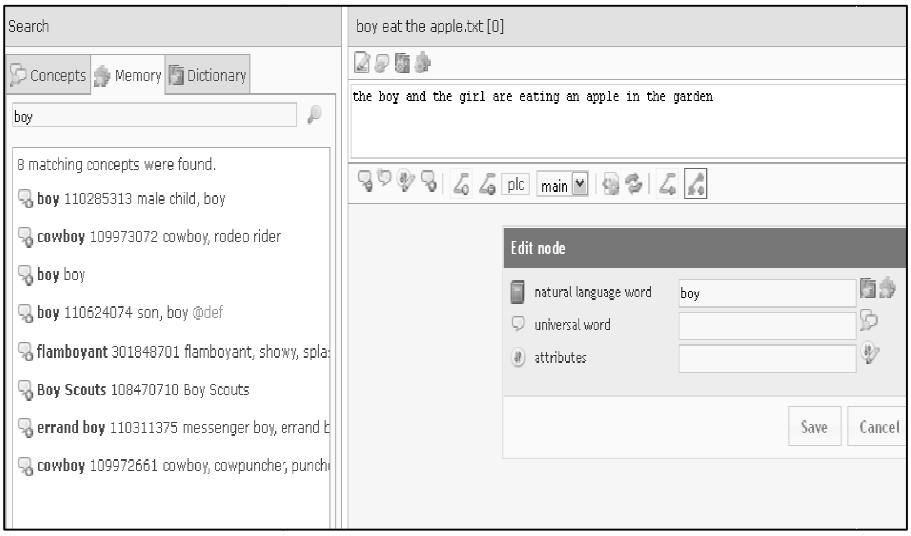

Figure 5: Memory tab

Dictionary tab

This tab offers much flexibility through providing the user with the option of using other dictionaries; the user can use the other dictionaries that exist in the another applications of the UNL web such as the dictionary of the EUGENE application or the dictionary of the IAN application or, he can upload his own dictionary provided that it conforms with the UNL dictionary format. This tab enables the user to create his own dictionary, thus creating the opportunity of having a specialized dictionary for specialized usage. Figure 6 shows the IAN application dictionary.

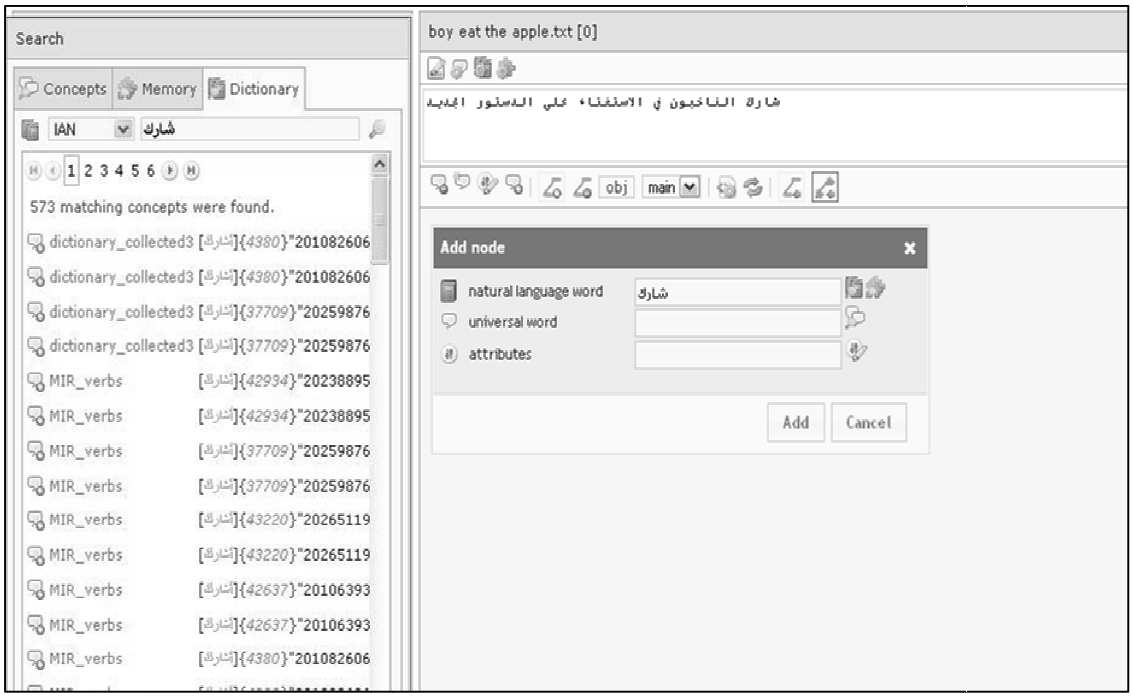

Figure 6: Dictionary tab

2) *Attributes Assignment:* After selecting the appropriate senses for each lexical item, some pieces of information will still be missing and need to be stated for each node in order to represent the whole meaning of the semantic network of the sentence. The UNL Editor provides a comprehensive set of attributes in order to convey these extra pieces of meaning, the added attributes have to be from the fixed list that the application has provided [22]. The process of adding the attributes is manual in the sense that the user of the tool has to add the attributes by writing them or, by coping them from a list of attributes that is available on the web site as in figure 7. The user can edit the added attributes through "modify UW attributes" button, the user can add or modify or delete any attribute. Furthermore, the UNL Editor has provided a special button for determining the entry called "entry assignment" button, since that the UNL specifications require that every sentence has to contain an entry node that represents the most prominent element in the sentence and that would be the starting point of the semantic graph.

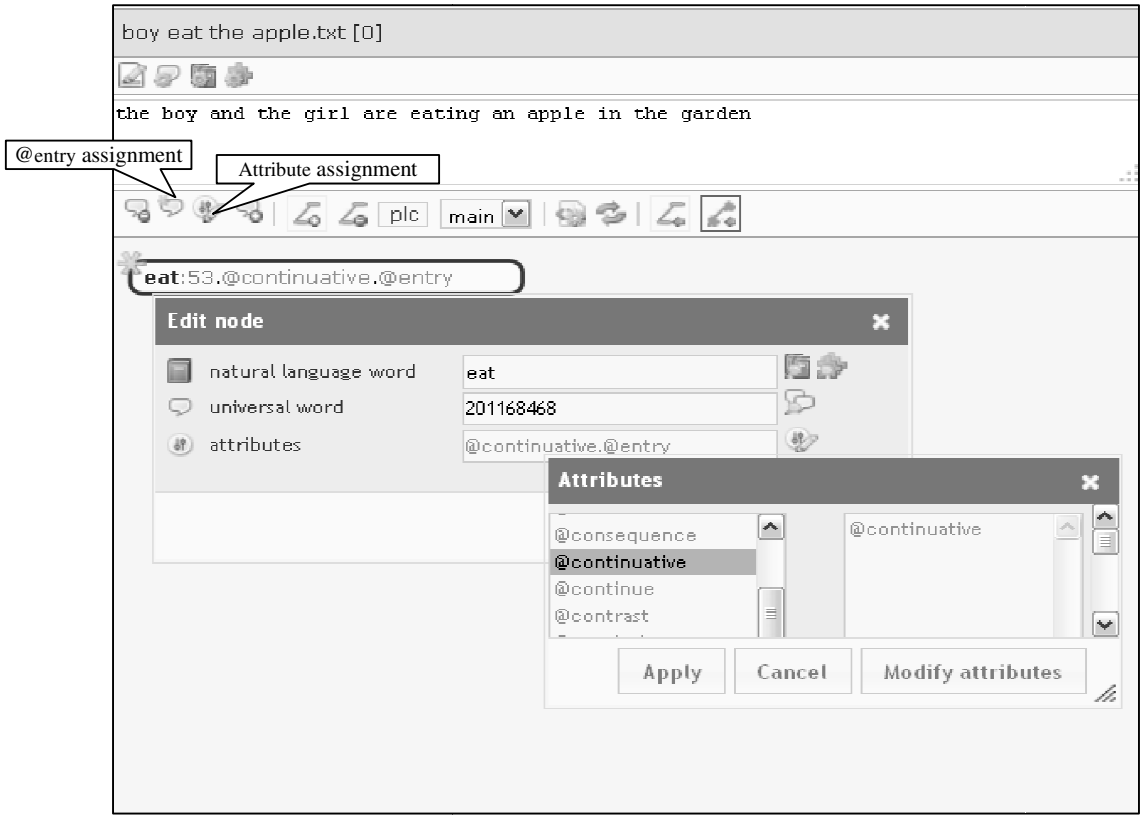

Figure 7: Assigning attributes

C. Linking the Nodes by Semantic Relations

The third and final step of analyzing a natural language sentence using the UNL Editor is the process of creating the semantic relations between the constituent of the sentence. Since that the UNL Editor is especially designed to offer the utmost appropriate environment for providing the analysis of natural language texts, it has provided a toolbar; including different buttons, that are necessary in performing all the needed operations in order to create the required semantic relations between the nodes, all of which is done through a graphic interface. For adding a relation between two nodes, the user could either click on "select relation", a button which consequently opens a list of all the semantic relations provided by the UNL framework from which the user can select the relation he finds most appropriate to convey the intended meaning or, the user could drag one of the two nodes he wants to choose a relation for onto the other where the set of UNL relations will appear and the user will be able to choose the suitable relation according to the meaning. Moreover, in order to modify a relation there has been another button called "remove selected relation" by which the user could remove the relation he selects through clicking on it and then clicking on the button. Every semantic relation used at the UNL framework has a specific direction; meaning that each relation should start from a specific node to go to another node in order to convey the meaning or otherwise the meaning could be distorted [23]. Therefore, a certain button has been provided to swap the direction of the relation after drawing it. It is called "swap selected relations nodes". The user could select the relation he wants to swap its direction by clicking on it then he could swap the relation by clicking on the button. This button has been designed with the intention of saving time and effort. Figure 8 shows a toolbar that includes all the buttons to create the semantic relations between the nodes.

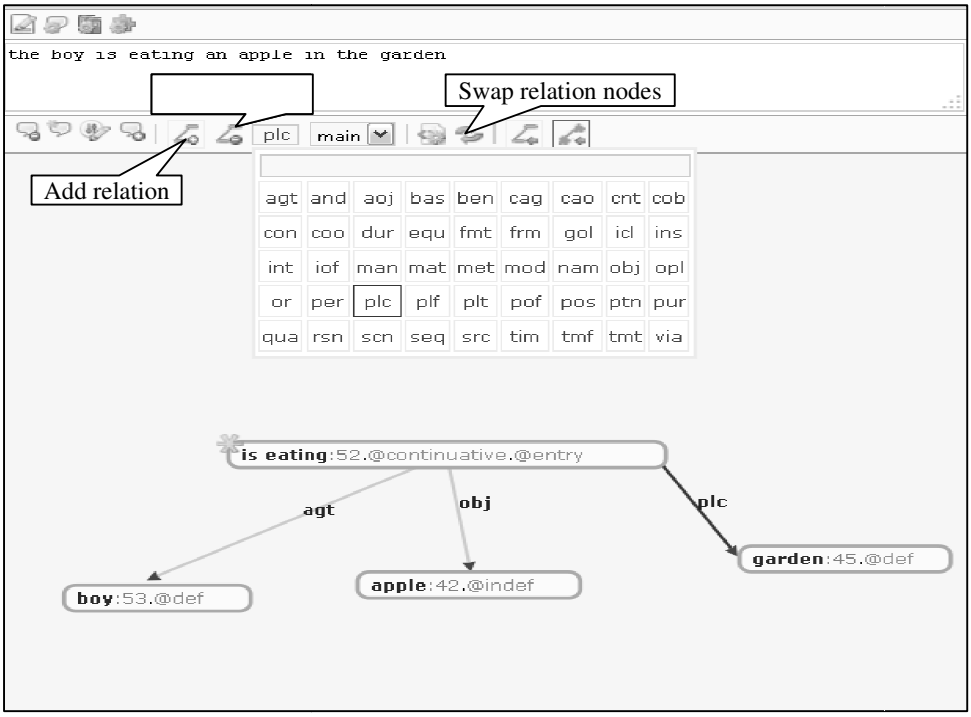

Figure 8: semantic relations between nodes

Creating the scope

The UNL representation is a hyper-graph, which means that it may consist of several interlinked or subordinate sub-graphs. These sub-graphs are represented as hyper-nodes which are named scope which roughly corresponds to the concept of dependent (subordinate) clauses. They are used to define the boundaries between complex semantic entities being represented. Scopes must be used to prevent semantic ambiguities in the following types of clauses:

1- adverbial clauses:

time: her father died (when she was young).
condition: (If they lose weight during an illness), they soon regain it afterwards.
purpose: They had to take some of his land (so that they could extend the churchyard).
reason: I couldn't feel anger against him (because I liked him too much).
consequence: My suitcase had become so damaged on the journey home (that the lid would not stay closed).
concession: I used to read a lot (although I don't get much time for books now).
place: He said he was happy (where he was).
manner: I was never allowed to do things (the way I wanted to do them).

2- adjective clauses:

The vegetables (that people often leave uneaten) are usually the most nutritious.

3- nominal clauses:

subjective: (Why you did that) is a mystery for me.
subjective complement: You can be (whomever you want).
objective: I know (that the weather will be very hot).

Every scope must contain one and only one attribute @entry, to be assigned to the head of the scope. The head of the scope is:

- The main verb, in verbal predicates;
- The subject complement, in nominal predicates;
- The head of the phrase, in phrases and non-finite clauses.

The user can create a scope by selecting the relation that will link the subordinate clause with the rest of the sentence and also selecting "new" from the "clause type" combo box . Then the scope will be created as a new node, as shown in Figure 9. A scope has been created as a new node with the name "01" and has been linked by the selected relation, as in figure 10. All the nodes inside the subordinate clause will be included in the scope, and the scope will be considered as a one unit or hyper node