NLUS – a Prolog Based Natural Language Understanding System

M.C. Lee, H.V. Leong

Dept of Computer Science, The Chinese University of Hong Kong, Shatin, Hong Kong
Dept of Computer Science, University of California, Santa Barbara, CA 93106, U.S.A.

Abstract

NLUS is a Prolog-based natural language understanding system, which exploits multi-knowledge representation formalisms. Sentences input from the user are converted into semantic networks and/or production rules; whereas the grammar rules are represented in predicate logic. After processing a sentence, the software may initiate its inference engine to deduce a response. A question would also be parsed and converted into a semantic structure which may contain possibly some missing information. To find an answer to a question, the missing information may be extracted from another related semantic structure in the working memory. At present, the software can handle sentences with relatively simple syntactic structures and some restricted types of questions. However, intelligent man-machine dialogues can be realized. It is particularly good at handling the rule-oriented type of sentences and is considered a decent prototype for experimenting with natural language understanding.

1. Introduction

Natural language understanding by a machine involves the process of transforming a sentence into an internal representation. In other words, it involves the process of making out the meaning of sentences in natural language and possibly storing the results and drawing inferences on the relationships depicted by the resulting representations. One obvious way to show if a computer understands an input sentence is to ask the machine questions about the sentence. If it can provide meaningful responses, the machine is said to understand the sentence.

Chomsky (1957) introduced the syntactic structures for natural languages, suggesting that a sentence is the surface appearance of a deep structure which represents the actual meaning of the sentence. A language like English could be generated by a generative grammar. In 1963, he proposed formal grammars for language study [2].

In 1970, Woods modified finite state transition networks into augmented transition networks (ATNS) which have been used extensively as a parsing method. ATNS were originally used to construct syntactic parsers; in the middle of the 70's, Woods modified them to incorporate the power of semantic analysis [8]. In 1979, Marcus started developing a deterministic parser, the Parsifil, which used a lookahead parsing method [6]. Knowledge is generally represented by means of semantic networks, production rules, predicate logic, scripts, and frames. Combinations of several kinds of representations are also possible. All these models and techniques are collectively known as the mathematical models.

There is another category of models, known as the psychological models based on linguistic theories. Schank developed his conceptual dependency theory which involves breaking down objects, actions and their relationships into elementary components [7].

The main problem of natural language processing is one of reproducing ordinary conversation in English. The job of the computer is to have all the necessary information to answer questions or to ask questions in ordinary English, or indeed for that matter, in any other natural language [4].

2. Structure of NLUS

NLUS contains primarily the modules: lexical analyzer; lexicon manager; parser; semantic network inference module; logical inference module; and response formulation module.

2.1 Lexical analyzer

The lexical analyzer accepts an input string from the user and sends the list of tokens generated to the parser. It reads the input sentence one character at a time, and converts the input sentence into a sequence of tokens. There are six different types of tokens recognized by the lexical analyzer: simple words, punctuations, possessives, proper names, short forms and quoted items. Simple words may be nouns, verbs, pronouns, adjectives, adverbs, prepositions, conjunctions, numerals, quantifiers, articles and interjections. Punctuations are symbols for comma, full stop, question mark, exclamation mark and semi-colon.

2.2 Lexicon manager

This module performs two functions: management of the lexicon, and learning any new word encountered by the system. The lexicon holds words known to the system. Its main function is to provide the required information so
that the system can parse and understand as much as possible a given statement. Words managed by the system are classified into 11 classes: noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, article, numeral, and quantifier. Each word has its attributes depending on its class. The semantic components synonyms and antonyms are connected by pointers so that it is easy to find the synonym or the antonym of a given word.

2.3 Parser

This module performs mainly the three functions: 1) To check if the list of tokens output by the lexical analyzer are syntactically valid; 2) To generate a parse tree; 3) To construct a semantic structure for an input sentence.

2.4 Semantic network inference module

This module finds answers, expressed in the form of a semantic structure, to a user question. The semantic structure of a question is similar to that of a statement except there is some information missing in the structure. It is not difficult to search for the missing information from the knowledge base.

2.5 Logical inference module

This module accepts the output from the Parser and converts the semantic tree into rules and facts. On the basis of the current rules and facts in the knowledge base, inference would be carried out, thus generating one or more new facts.

2.6 Response formulation module

This module uses the information acquired from either the semantic network inference module or the logical inference module to formulate a response to a user input. The response presented to the user should be in the form of a complete English sentence.

3. Knowledge representation

In general, there is no single knowledge representation formalism which suits all kinds of applications. For the present application, four kinds of structures are considered essential for representing the different types of knowledge processed by NLUS.

3.1 Phrase structure grammar rules

The knowledge for directing the parsing process is expressed in the form of phrase structure grammar rules. Presented below are some simple phrase structure grammar rules.

\[<\text{Sentence}> \rightarrow <\text{Noun Phrase}> <\text{Verb Phrase}>\]
\[<\text{Noun Phrase}> \rightarrow <\text{Article}> <\text{Noun}>\]
\[<\text{Verb Phrase}> \rightarrow <\text{Verb}> <\text{Noun Phrase}>\]
\[<\text{Noun}> \rightarrow \text{cat} | \text{dog} | \text{cow} | \ldots\]
\[<\text{Verb}> \rightarrow \text{eat} | \text{drink} | \text{learn} | \ldots\]

3.2 Parse tree

Upon parsing a given input sentence, the parser produces a parse tree which indicates explicitly the syntactic structure and also, to some extent, the meaning of the sentence. Sometimes, a sentence may have different parse trees, implying its meaning being ambiguous. One way of dealing ambiguity is to let the parser generate the parse trees one by one until a meaningful response can be produced by the system. The other solution is to allow the parser to ask the user if a given parse tree corresponds to the intended interpretation; if not, another parse tree would be generated.

3.3 Semantic networks

While generating a parse tree, the parser produces a knowledge structure often known as a semantic network. Within such a structure, items are connected together by means of various relationships. The system establishes the network on a sentence by sentence basis. The main verb of a sentence is made the centre of the corresponding semantic structure. An example is shown in Figure 1a. The sentence information pointer points to a list of attributes about the sentence, including the tense of the verb, the adverbial information, an indicator to denote if the sentence is a question, and a flag to indicate whether the sentence is in a positive or a negative form. The list also includes some terms qualifying the verb. Objects in this world are called entities. There are terms qualifying the entities and they may belong to the semantic categories like location, quantity, feeling etc. If a syntactic role is absent, the corresponding pointer would be set nil. Figure 1b shows an example where the pointer for the indirect object is set to nil.

![Figure 1 - Simplified Semantic Structures](image-url)
The semantic structure of a sentence is represented in Prolog according to the format:

```
verb(sentence information, subject, object, indirect object).
```

For instance, the structure for the sentence "John gives Mary a book" would be represented as follows:

```
give([tense(present)], subject([ent, [John, null]], object([ent, [book, [null]]]), ind_object([ent, [Mary, null]])]
```

The main verb, in its infinitive form, of a sentence constitutes the functor of the structure. The first argument is a list containing the information of the sentence. To indicate that a sentence is a question, the atom `quest` will be added to this list. The atom `neg` will be added to this list if the sentence is in negative form. Any terms qualifying the verb would also be added to this list. The second argument, a structure for the subject(s) of the sentence, has the format:

```
subject([flag, [subject name, [subject name, [descriptors]], [descriptors]], ' '])
```

The `flag` indicates if there is one or more subjects in the sentence. If the flag is `ent`, there will be only one subject. If the flag is `and`, there will be more than one subject. If the flag is `or`, any one of the subject names can be a subject. Following the flag is one or more lists containing the subject information. There are two parameters in each list: the subject name, and the descriptor. The descriptor describes the subject.

The third argument is for the object and the fourth argument for the indirect object.

The representation of a question is similar to that of a sentence. For a question, the atom `quest` is added to the first argument of the semantic structure. The missing information in the question is replaced by the structure with the functor `unknown`. For example, the structure for the question "Who sailed the boat?" will be organized as:

```
sail([quest, tense(present)], unknown([[]]), object([ent, [boat, [art(definite)]]]), nil).
```

The logical inference module converts the semantic tree into another representation known as production rules. A production rule has the general format:

```
IF condition THEN conclusion
```

where `condition` is a conjunction of one or more premises and the `conclusion` may contain one or more facts. When the `condition` is true, the fact(s) for the `conclusion` part would be generated.

As an example, the sentences `all men are mortal` and `Mary is a girl` can be expressed in the form of a rule and a fact respectively:

```
be(X, man) -> be(X, mortal)
be(Mary, girl)
```

4. Sample dialogues

In this section, two sample dialogues are presented to show the types of questions which NLUS can handle, where the user inputs are in italic phase. The questions posed in the dialogue serve to reflect, to some extent, the intelligence of the system.

```
** John and Mary visited the zoo on Monday.
** ok
** She saw a tiger in the cage.
** ok
** He saw a tiger.
** ok
** When did John visit the zoo?
** John visited the zoo on Monday.
** What did she visit?
** Mary visited the zoo.
** Who visited the zoo?
** John, Mary visited the zoo.
** Did John visit the zoo?
** Sorry, I cannot parse your sentence.
** Did John visit the zoo?
** Yes, John visited the zoo.
** Who saw the tiger?
** John, Mary saw the tiger.
** Where was the tiger?
** The tiger was in the cage.
** What did he see?
** John saw the elephant, the tiger.
** Did Peter see the tiger?
** No, I cannot find or deduce such a fact.
```

4.1 Dialogue 1

The dialogue is as shown in Figure 2. The answers to the questions posed in this dialogue reflect that NLUS is able to extract the semantic meanings of a sentence. In addition, the system is able to accumulate knowledge in the knowledge base and to answer questions by combining the semantic meanings of the relevant sentences input by the user.

4.2 Dialogue 2

While the answers to the questions in dialogue 1 can be extracted by the semantic network inference module, those to the questions in dialogue 2 shown in Figure 3 have to be deduced by the logical inference module. Dialogue 3 reflects NLUS having the ability to convert some English sentences into production rule representations.
**Everybody is clever.**
**John is stupid.**
**I can deduce that John is not a man; the reason is:**
If X is not clever then X is not a man.
Now, John is not clever.
**Peter is a man.**
**I can deduce that Peter is clever; the reason is:**
**Is Mary clever?**
**Yes. Mary is clever.**
**Everybody is mortal.**
**You told me something similar to this before!**
**All human are stupid.**
**You told me before the rule:**
If X is human then X is clever.
But now you say something opposite:
If X is human then X is stupid.
**Some men are fat.**
**Peter may be fat; the reason is:**
If X is a man then X may be fat.
Now, Peter is a man.
**An elephant is bigger than a monkey.**
**ok**
**A lion is bigger than a monkey.**
**ok**
**Is an elephant bigger than a monkey?**
**Yes, an elephant is bigger than a monkey.**

Figure 3 – Dialogue 2

5. Concluding remarks

NLUS has been implemented with Arity Prolog on the IBM PC compatible and equipped with the features mentioned in this paper.

The dialogues presented in the foregoing sections demonstrate that the system can capture the semantic meanings of certain English sentences and can handle some types of questions. However, the system can only parse sentences having relatively simple grammars. In addition, the questions have to be structured in a restricted way. That is, it may not be able to answer a simple question if paraphrased differently. For instance, though it can answer a question like "Did John visit the zoo on Monday?", it cannot answer a similar question "What did John do on Monday?". Therefore, the system should be enhanced to cater for more question patterns.

The types of sentences which can be parsed are also limited. To amplify its parsing capability, grammar rules for more complex sentence patterns have to be introduced. This may not be an easy task. In particular, to program the machine to parse and to understand the various propositional phrases can be time-consuming. In addition, the parser can be modified to make it more forgiving in handling grammatical errors.

The lexicon has captured only a limited number of words. In order to improve the practicality and the power of the system, more words must be included in the lexicon. Moreover, the response or the answer generated by the machine could be presented in more natural English. It is envisaged that to improve the system in this direction would not be a complicated task.

NLUS was first implemented on the 8086-based IBM PC compatible by two final year computer science project students in 1987. Since parsing a sentence often took a long time, much time was spent on testing the implementation. As a result, it was not possible to add on the grammar rules for the more complex sentence patterns. Indeed, the most obvious drawback of the software was its slow response. NLUS has now been ported to an 80386-based PC, resulting in a substantial improvement in response time. As a result, NLUS has become a more interesting software for experimenting with natural language understanding. Obviously, it is vital to develop this type of system on a high-speed machine. We plan to make the above-mentioned enhancements to the system so that it can understand more sentence and question patterns and can carry out more interesting man-machine dialogues.

7. References