NLOMJ--NATURAL LANGUAGE OBJECT MODEL IN JAVA

Jiyou Jia(1), Youfu Ye(2), Klaus Mainzer(3)
(1)(3) Institute for Interdisciplinary Informatics, University of Augsburg, Germany
(2) Information Technology Designing & Consulting Institute, China

Abstract: In this paper we present NLOMJ—a natural language object model in Java with English as the experiment language. It describes the grammar elements of any permissible expression in a natural language and their complicated relations with each other with the concept "Object" in OOP. Directly mapped to the syntax and semantics of the natural language, it can be used in information retrieval as a linguistic method. Around the UML diagram of the NLOMJ the important classes (Sentence, Clause and Phrase) and their sub classes are introduced and their syntactic and semantic meanings are explained.

Key words: natural language processing (NLP), object oriented programming (OOP), natural language object model in Java (NLOMJ)

1. BACKGROUND

We have developed an innovative web-based human-computer-interaction system with natural language for foreign language learning: CSIEC (Computer Simulator in Educational Communication) [1]. The kernel of this system is the natural language understanding mechanism (NLML, NLOMJ and NLDB) and the communicational response (CR). NLML (Natural Language Markup Language) is a markup language to describe the grammar of an expression in a natural language. It is produced to an expression of this natural language by a parser written according to the grammar rules and lexicon of this language [2]. We use English as the experiment language in our system.

With the simple structure of the markup language the NLML can be easily parsed by an OOP language to construct the object model of the nodes in the markup language. As for the NLML the nodes are just the grammar elements of the natural language. So we have selected Java, the typical OOP language, to parse the NLML and to represent the grammar elements and their relations with the concept of Object. We call this technique NLOMJ. It enables us to extract the information we need from any permissible expression of a natural language, hence can be used as a linguistic method in intelligent information processing.

The UML diagram for the whole NLOMJ is shown in figure 1. In the following paragraphs we introduce the content of NLOMJ around this diagram in the order of sentence, clause and phrase very structurally.

2. SENTENCES

2.1 Super class "Sentence" and Interface "Sentence_operation"

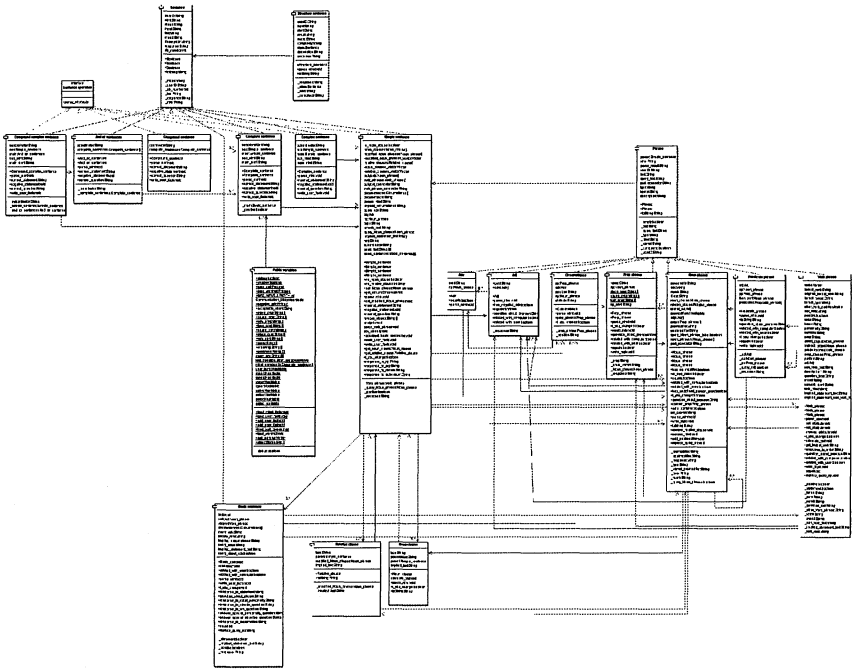

Figure 1. UML diagram for NLOMJ

All sentences with diverse complexity, tense, mood and voice must have at least one subject and one verb phrase. So we use the class "Sentence" to generalize the common features of the sentences. Because there are great structure differences for the sentences with different complexity, the compound complex sentence, compound sentence, complex sentence and simple sentence are all treated as the subclass of the class "Sentence".

The attributes in the super class "Sentence" include "mood", "input", etc, which are all with the type "String". The "mood" represents the mood of the sentence therefore may have one of the values--"statement", "question", "order", etc. The attribute "input" is the input from the user, whereas the "text" is inferred by the parser from the "input" and other attributes. So the "input" does not always equal the "text". The "nlml" is the parsing result in NLML format. The "description" is the syntactic and semantic description of the "text".

In order to get the object model from the NLMLs a method for the parsing of the NLML should be implemented in every subclass of the "sentence". So an Interface "Sentence_operation" is written with only this method "parse_nlml".

2.2 Sentences with different complexity

The "Compound_complex_sentence" consists of a subordinate "Simple_ sentence" and an "And_or_sentence". The attribute "subordinator" with the type "String", the "sub" with the type "Simple_sentence", and the "main" with the type "And_or_sentence" represent the subordinator, the subordinate sentence, and the main sentence, respectively. The key is the semantic decomposition of the compound complex sentence into simple sentences, e.g. the compound complex sentence with the conjunction "and" connecting N main clauses can be decomposed into N complex sentences, each of which has the same subordinate clause.

The "And_or_sentence" consists of a coordinator ("and", or "or") represented by the attribute "coordinator" and several instances of the class "Complete_sentence" represented by the array "complete_sentences".

The "Complete_sentence" consists of a subordinator, a subordinate clause and a main sentence represented by the attribute "subordinator", by the sub" with the type "Simple_sentence", and by the "main" with the type "Simple_sentence", respectively. If the subordinator of an instance is null, the "sub" is meaningless, and this instance is essentially the same as an instance of "Simple_sentence". The array "complete_sentences" in the class "Complex_compound_sentence" and the "Compound_sentence" uses this special case. But if either the "subordinator" or the "sub" is null, this instance is the same as an instance of the class "Complex_sentence". In other

words the "Complete_sentence" is either a "Complex_sentence" or a "Simple_sentence". It does not exist in the real language, however is useful to describe the personality of the "chatting robot" in the "CSIEC" project, as some personalities can be stated by simple statement sentences as the facts, whereas others are states or behaviors happening only with some conditions, which should be represented by the "Complex_sentence".

The "Compound_sentence" consists of the conjunctions represented by the attribute "coordinator" and several complete sentences represented by the array "complete_sentences".

The "Complex_sentence" consists of a subordinator, a subordinate clause and a main sentence. Here the subordinator can not be null.

Summarily the simple sentence is the most elementary sentence and the end point of the parsing of all sentences with different complexities.

2.3 Simple sentence

This kind of sentence has at least one verb phrase. It may contain relative clause and noun clause which contain also verb phrases. So we deal with them at first, i.e., search the tag "noun_clause" and "relative_clause" and replace their contents with a noun phrase and a preposition phrase, respectively, then parse every noun clause and relative clause, and save them in the arrays. In the UML diagram the blue arrow with the number "0...." from the class "Simple_sentence" to the "Noun_clause" and to the "Relative_clause" indicates this relation.

We should think over the parsing with different moods. For example, if the mood is "np", "what terse exclamation", or "about", the sentence treats essentially one noun phrase, therefore can be parsed as a noun phrase. The attribute "np" in the class "Simple_sentence" stands for it.

The sentence may have several subjects. They are all represented by instances of the class "Noun_phrase" and saved in the array "subjects". In order sentences and some clauses like infinitive, gerund, etc, there may be a negative phrase like "don't", "not". So it should be parsed firstly via the tag "neg". Finally we get the verb phrase via the tag "verb_phrase". If the content of the tag "verb_phrase_connector" is null, there is only one verb phrase; otherwise there are several which can be sequentially got via the tags "verb_phrase_part". They are put into the array "verb_phrases".

The key point is how to combine these subjects, verb phrases and circumstances. The conjunction connecting the noun parts and the conjunction connecting the verb phrases are: "and", "or", "neither...nor". It is common that either the subject or the verb phrase is made up of several parts, e.g., N parts. In this case this simple sentence can be decomposed into N basic sentences. If the conjunction is "and", these N basic sentences are

independent; if it is "or", the relation among these basic sentences is single choice; if it is "neither_nor", we can form N independent basic sentences through negative operation. The circumstances are suitable for every basic sentence.

The method in the "Simple_sentence", "contruct_basic_sentences()", implements this operation of combination. Then the basic sentences are stored into the two dimensional array "basic_sentences". The values of the attribute "text" and "description" are also calculated according to the "mood" and the element values of "basic_sentences". The concept "basic sentence" used here represents the sentence with no more than one subject phrase and one verb phrase. It is represented by the class "Basic_sentence", which is just a subclass of the class "Sentence", as its instance is not formed by parsing the NLML, but constructed by the given subject phrase, verb phrase, and other parameters.

3. CLAUSES

In NLOMJ the relative clause and noun clause are defined as subclass of the "Simple_sentence", for they either are full simple sentences or can be extended to semantically equal simple sentences. The attributes and operations in the "Simple_sentence", such as "text", "subjects", "verb_phrases", "construct_basic_sentences", etc, can be still applied in the two subclasses. Additionally some special attributes and operations are needed.

3.1 Noun clause

There are several types of noun clauses:

"that", "whether", "whether or not": the noun clause consists of a preceding word and then a simple statement sentence without noun clause. The "implied_text" equals the "text".

"query clause": the implied text is a question and can be obtained with the same method as that used in obtaining the "text" in the Basic_sentence with the mood "question". The key is how to get the auxiliary verb and the other parts in the verb phrase, which is realized in the class "Verb_phrase".

"query_to": in this type there is no subject, so it is more difficult to infer the implied question. If the noun clause is the subject of the main sentence, this question may be suitable to all people. In this situation we assign "*a person*" as the subject of the question. If the noun clause is the object in the main sentence or the object in a prepositional phrase, we can assign the subject in the main sentence as the subject of the noun clause.

"normal_to": there is also no subject in the noun clause. If this type of infinitive appears as the object of some special verbs, which are shown in the various construction forms of the verb phrase, we don't have to obtain the implied statement with great efforts, because the subject of the infinitive clause is the same as the subject in the main sentence and the whole verb phrase including both the main verb word and the infinitive clause expresses a mental state of the subject and the semantics can not be expressed by the main verb word or the infinitive clause alone.

3.2 Relative clause

The relative clause is a full relative clause if it has a subject; otherwise it is a terse relative clause, whose actual subject is evidently the noun phrase it modifies. The full relative clause starts with a query word (if there is none it is "which") which points to the modified noun phrase in front of it. So we can infer the implied statement sentence by two steps: firstly get the statement sentence with the query noun phrase in the way we use in the noun clause, secondly replace the query word with the modified noun phrase.

For the terse sentence it is difficult to set the tense of the implied statement sentence. If it is the present participle, the implied statement sentence may have the tense "progressive". If it is the past participle, the implied statement sentence may have the tense perfect or past. This character is labeled by the tag "voice" with the value "passive". If it is passive infinitive clause, we can set the tense with the value "future" or use the model verb "should".

4. PHRASES

4.1 Super class "Phrase"

The sentence consists of various phrases, so at the end of the parsing of the sentences comes the parsing of the phrases. On other side the phrase exists in the sentence and has its syntax und semantic function in the context of its existing sentence. The super class "Phrase" in NLOMJ represents its common feathers. Its attributes include the "nlml", "text", and "description" which are similar with those in the class "Sentence". Besides it has also the following ones: the "parent" with the type "Simple_sentence" representing the simple sentence where this phrase exists; the "part_connector" with the type "String" representing the conjunction word if this phrase consists of

several parts; the "kernel" with the type "String" representing the kernel word in the phrase; etc.

Every subclass of the "Phrase" has some special attributes and its own method "parse_nlml()" to parse the "nlml" in order to form an instance of this subclass. Now we introduce the subclasses of the "Phrase".

4.2 Adverb

The class "Adv" represents the adverb and has two attributes: "grad" and "np". The string "grad" represents the grad of the adverb and is obtained via the tag "grad", i.e. "abso" (absolute), "comp" (comparative), or "supl" (superlative). The "np" with the type "Noun_phrase" is obtained through the parsing of the other parts in the NLML, if the type of this adverb is "so_that", "so_as", "enough_to", "too_to", or "adv_than", because these types of sentences express the result or the extent of the adverb with a compared noun phrase, a statement sentence or a noun clause (infinitive clause), all of which can be parsed with the class "Noun_phrase".

4.3 Adjective

The class "Adj" represents the adjective. It has two attributes: "grad" and "advs". The "grad" is similar with that in "Adv". The array "advs", whose elements are instances of the class "Adv", represents the adverbs modifying it.

4.4 Circumstance

The class "Circumstance" represents the circumstance. It can be adverb, prepositional phrase or noun clause (infinitive clause or participle clause), what is decided by the attribute "type". Its attribute "position" represents the position of the circumstance in the sentence: pre, mid or post, and the "attribute" represents its semantic function in the whole sentence: place, time, way, and others.

4.5 Prepositional phrase

The class "Prep_phrase" represents the prepositional phrase. It has the attribute "prep" with the type "String" representing the preposition word and the attribute "np" with the type "Noun_phrase" representing its object.

4.6 Noun phrase

The "Noun_phrase" represents the noun phrase and has many special attributes such as "personality", "number", "case", etc. It may have several parts connected by conjunction and every part can be in the normal form: pre modifiers + kernel + post modifiers. The kernel can have different types, like "countable noun", "number", etc, what is decided by the attribute "type". The pre modifiers can be adjective, article, determiner, quantifier, etc. The post modifier can be prepositional phrase or relative clause. The noun clause as a noun phrase is specially considered.

4.7 Predicate phrase

The "Predicate_phrase" represents the predicate which may be adjective, noun phrase or prepositional phrase, what is encoded in the attribute "type". For the adjective a noun phrase or noun clause can be attached by "as...as", "than", "too...to", "enough...to", or "so...that" to express the compared object, the extent or the result of the adjective.

4.8 Verb phrase

The class "Verb_phrase" represents the verb phrase and is the most important and complicated phrase in constructing a sentence and may consist of all other types of phrases. It describes the actions or the states of the subject, or the subject's relation with other things or persons which can be described by direct and (or) indirect objects represented by the two instances of "Noun_phrase" ("direct_object" and "indirect_object"), or predicates represented by the instances of "Predicate_phrase" ("predicate"). The concrete pattern of the verb phrase is decided by the attribute "verb_type". In the parsing firstly the verbal attributes are obtained, such as "personality", "number", "voice", "tense", "kernel_tense", "verb_type", etc. Then the other parts in the verb phrase are obtained corresponding to the "verb_type". At last the verb words are obtained successively.

REFERENCES

1. Jia, J. CSIEC (Computer Simulator in Educational Communication): An Intelligent Web-Based Teaching System for Foreign Language Learning. In Proceedings of ED-MEDIA04 (17th World Conference on Educational Multimedia, Hypermedia and Telecommunications), P. 4147-4154.

2. Jia, J. NLML--a Markup Language to Describe the Unlimited English Grammar. Submitted to German conference for artificial intelligence 2004. Also available at http://arxiv.org/abs/cs.CL/0404018.