International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 3– No.7, August 2012 – www.ijais.org

Processing Natural Language Requirement to Extract Basic Elements of a Class

Poonam R. Kothari
Master of Computer Engineering
Pune Institute of Computer Technology, Pune University, Pune, India

ABSTRACT

This paper presents the efficient way to obtain basic elements of a class diagram from natural language (NL) requirements. User provides the requirements in simple English in paragraph and the designed tool i.e. NLPC (Natural language Processing for Class) applies natural language processing (NLP) methods to analyze the given input. NL text is semantically analyzed to obtain classes, data members and member functions. NLPC helps to fill the gap between the informal natural language used to describe problems and the formal modeling languages used to specify software solutions. Input to this tool is clearly specified user requirement. With correct inputs, NLPC undergoes stages like Preprocessing, Part of Speech (POS) Tagging, Class Identification, Attribute and Function identification and then plotting the classes.

Keywords

Natural Language (NL),Natural Language Processing (NLP), Part of Speech (POS) tagging,

1. INTRODUCTION

The idea behind this project is to develop a tool which will extract the basic elements necessary for creating a class diagram from clearly specified user requirements. NLPC successfully extracts classes, its data members and member functions from the given input.

Class diagram helps software developers to identify the key classes from the given input and to identify the functions through which the classes interact with each other [1][2][3]. To plot class diagram from the user requirements becomes time consuming. The motivation behind the development of NLPC is to automate the plotting of class diagram from user requirements, and hence to save the time taken by designer to plot the diagram. NLPC achieves this by using WordNet 2.1 [4] more efficiently.

NLPC is helpful for Software Designers, Software Developers and Software Engineering students. Figure 1 describes the Workflow diagram of core capabilities of NLPC. Input to NLPC is user requirement with detailed specifications.

This input is then processed through different stages like Preprocessing, POS Tagging, Class Identification, Attribute and Function Identification and finally plotting Classes.

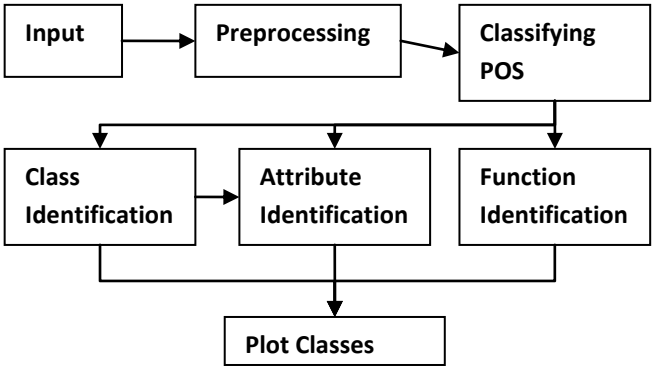

Figure 1 Core Capabilities of NLPC

2. RELATED WORK

Object Oriented Analysis (OOA) exhibits some unique difficulties that are inherited form NL. Firstly, NL is highly informal in nature, with speakers or writers inventing new forms and combinations of language. Indeed, sentence patterns of NL can be complex and ambiguous, which may lead to multiple interpretations [5][6]. Natural Language Processing (NLP) in Object Oriented Analysis (OOA)can lead to automatic text analysis. Therefore, NLP can disambiguate NL descriptions and assist model generation.

LIDA (Linguistic assistant for Domain Analysis) [7] methodology started with text description in which analyst reads and marks nouns and verbs then determines the suitable objects and function.

According to Rada Mihalcea, Hugo Liu, and Henry Lieberman [8], Natural Language Processing (NLP) holds great promise for making computer interfaces that are easier to use for people, since people will be able to interact with the computer in their own language, rather than learn a specialized language of computer commands. From design and coding point of view it is very important to identify the basic elements of object oriented programming. Also for beginners of object oriented programming it is difficult to identify class, data members of the class, and member functions from generalized problem definition. A new creative method should be prepared to help them to handle the problem.

According to Mohd Ibrahim, Rodina Ahmad [9], "Requirements Analysis and Class Diagram Extraction (RACE)" is a desktop instrument which will assist requirement analysts and software engineering students in analyzing textual requirements, finding core concepts and

their relationships, and step by step extraction of the class diagram. The evaluation of RACE system is in process.

Khalid Daghameen and Nabil Armana [10] developed a tool to implement a system that automates the building of class diagrams from free-text. They first applied natural language processing techniques for understanding the written requirements, and then used domain knowledge represented by domain ontology to improve the performance of class identification. However, their approach has the overhead of handling the domain ontology to identify classes of the class diagram. An important issue is that there is no tool that automates the process between requirement analysis and design phases. Their tool introduces a semi-automated process.

Mich L. [11] proposes a NLP system, LOLITA to generate an object model automatically from natural language requirement. It considers nouns as objects and it uses links to find relationships between objects. LOLITA system does not distinguish between classes, attributes, and objects. This approach is limited to extract objects and cannot identify classes.

Zhou and Zhou [12] propose a methodology that uses NLP and domain ontology. It is based on that the core classes are always semantically connected to each other"s by one to one, one to many, or many to many relationships in the domain. This methodology finds candidate classes using NLP through a part of speech (POS) tagger, a link grammar parser, linguistic patterns and parallel structure, and then the domain ontology is used to refine the result [12].

Unlike LIDA, LOLITA, and the tool developed by Khalid Daghameen and Nabil Armana, NLPC is a fully automated tool. Also, the output of RACE does not show functions, whereas NLPC identifies functions and data members too. Further it can be developed to identify relationships between classes.

3. METHODOLOGY

NLPC accepts clearly specified user requirement as an input. The given input goes through following stages:

3.1 Preprocessing

Preprocessing is, breaking down the sentences into set of words. The stop words are then identified and removed from the set of words. The remaining words are then further processed

3.2 Classifying Parts of Speech (POS)

The next stage is classification [13] of word into three main types, viz, noun, verb or adjective using WordNet 2.1.

3.3 Class Identification

Nouns are very complex and can play multiple roles [14]. It can be categorized either as class [15] or as data member. Rules are applied on the identified nouns and then they are categorized as class and the rejected nouns can be categorized as data members.

Class identification rules are as below:

1. If hypernyms of identified noun describes it as a "God", then discard it.
2. If the identified noun occurs only once in the input and its frequency is less than 2%, then discard it [9].
3. If the identified noun is related to design element or location name, then discard it [14].
4. Otherwise consider it as a class [9].

Identified classes are stored in a separately.

3.4 Attribute Identification

Adjectives and nouns which are not classes are the key candidates for attribute identification.

Identify the class which belongs to the same statement, check the correspondence, and store the attribute and corresponding class.

3.5 Function Identification

Verbs are the key candidates for function identification [15]. Also if the verbs are connected by "_" (underscore), then consider them as a candidate for member function identification [9].

1. Check whether the identified verb is from predefined function list.
2. If it is not a part of predefined function list then identify the class which belongs to the same statement (as that of identified verb).

Identified member functions are stored along with corresponding class.

4. ALGORITHM

Step 1. Accept the clearly specified requirement.

Step 2. Perform tokenization on the user input.

Step 3. Identify stop words and remove them from the input.

Step 4. Send each word to Word Net 2.1 to find its type (Noun/ verb/ adjective).

Step 5. Store each word in the concept list (hash table) along with its identified type.

Step 6. Generate the hypernyms list for each noun using Word net 2.1 and store it in individual concept list.

Step 7. Identify classes by processing each noun (ref Class Identification).

Step 8. Identify member functions by processing each verb (ref Function Identification).

Step 9. Identify data members by processing each adjective and non-class nouns (ref Attribute Identification).

Step 10. Integrate data obtained from step 7 to 9 and plot the classes.

5. Example

Suppose a user wants to develop a system which keeps track of record of a shoe company. He would give the requirement

to the Requirements Engineer in natural language. The Requirements Engineer would then put this requirement in standard grammatical format and use the NLPC to generate the classes. Following is an example of the requirement:

Consider a Shoe_Company. Customer needs to_register with the Shoe_Company. Customer can place order. Shoe_Company will dispatch_the_order. For registration, Customer need to provide_basic_details, and payment_details. Customer also needs to provide shoe_sizes, gender, and any special_details. Shoe_company makes different shoe_types. Shoe_company will verify_the_details of an order. Then Shoe_company will process an order. An order can be waiting_for_dispatch. An order can be waiting_for_delivery to the customer waiting_for_credit, waiting_for_clearance.

Table 1 Expected result of the above example

Expected classes	Expected Attributes	Expected Functions
Shoe_company	shoe_types	dispatch_the_order verify details process order
Customer	shoe_sizes gender special_details	to_register make_order provide_basic_details place_shoe_order
Order		waiting_for_dispatch waiting_for_delivery waiting_for_credit waiting_for_clearance

Figure 2 shows the output of NLPC for above mentioned example of Shoe Company.

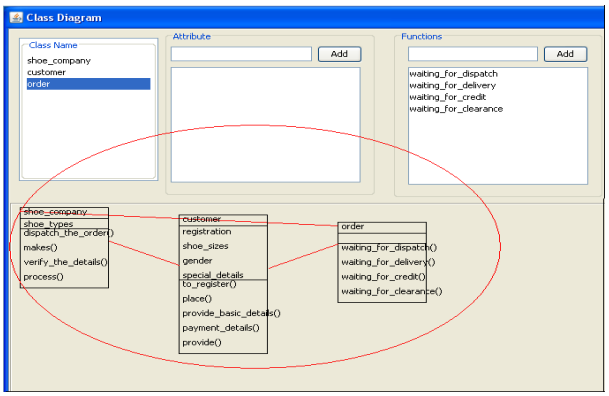

Figure 2 Experimental result obtained from NLPC

Class diagram (marked in red circle in Figure 2) is as follows:

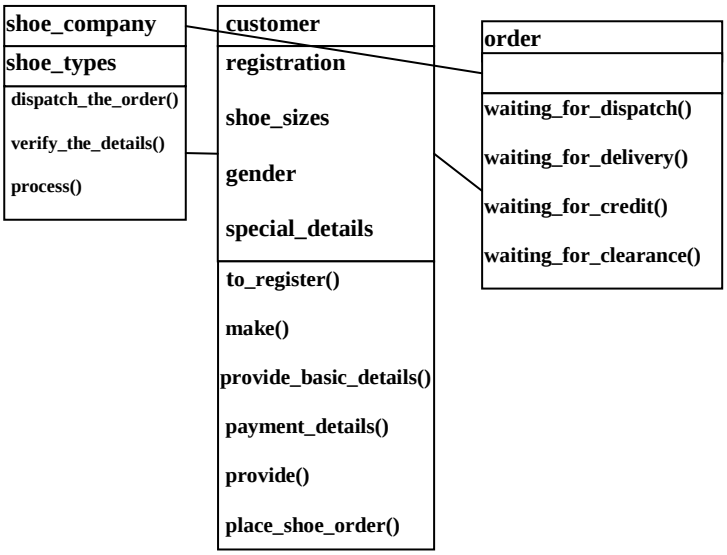

Figure 3 Class Diagram obtained from NLPC

The accuracy of NLPC can be checked from comparing result of NLPC against expected result.

6. CONCLUSION

NLPC is useful for Software Designers or Requirement Engineers, who give the natural language requirement and readily get the classes with all its functionalities. This classes act as a base for UML diagrams.

To get the exact output from NLPC, it is important to give the requirement in detail; with the sentence structure being in standard grammatical format.

Developers can use NLPC to identify the key classes and their functionality, which is an important part of software development.

Engineering students can use NLPC to study class diagrams. Class diagrams are helpful to learn and understand software languages, such as C++ and JAVA.

7. REFERENCES

[1] Hans_Erik Erikson, Magnus Penker, Brian Lyons, David Fado, "UML 2 Toolkit" Wiley Publishing.

[2] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Object-oriented Modeling and Design*, Prentice Hall.

[3] Meyer, B. (1997). Object-Oriented Software Construction. Prentice Hall.

[4] G.A. Miller, "WordNet2.1," 2006; http://wordnet.princeton.edu/.

[5] N. Boyd, "Using Natural Language in Software Development", Journal of Object Oriented Programming, Feb. 1999.

[6] M. Osborne, C.K. MacNish, "Processing Natural Language Software Requirement Specifications", Proceedings of the 2th International Conference on

Requirements Engineering, IEEE, 15-18 April 1996, pp. 229-236

[7] Overmyer , S.P, Lavoie, B, Rambow,O. 2001.Conceptual Modeling through Linguistic Analysis Using LIDA. IEEE.

[8] Rada Mihalcea, Hugo Liu, and Henry Lieberman,"NLP (Natural Language Processing) for NLP (Natural Language Programming)" pp. 319–330, 2006.

[9] Mohd Ibrahim, Rodina Ahmed, "Class diagram extraction from textual requirements using Natural language processing (NLP) techniques," Proceedings of Second International Conference on Computer Research and Development, pp. 200-204, 2010 IEEE.

[10] Khalid Daghameen, Nabil Arman, "REQUIREMENTS BASED STATIC CLASS DIAGRAM CONSTRUCTOE (SCDC) CASE TOOL," Journal of theoretical & Applied Information Technology, Islamabad Pakistan, pp. 108-114,may 2010.

[11] L. Mich, NL-OOPs: "From Natural Language to Object Oriented Using the Natural Language Processing System LOLITA.", Natural Language Engineering, 1996, pp.161-187.

[12] Xiaohua Zhou and Nan Zhou, 2004, Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology.

[13] Deva Kumar Deeptimahanti,Muhammad Ali Babar, "Automated tool for generating UML models from Natural Language Requirements," International Conference on Automated Software Engineering, IEEE,2009, pp 680-682.

[14] Ke Li, R.G.Dewar, R.J.Pooley, "Requirements capture in natural language problem Statements ," 2003.

[15] Pressman,"Software engineering", A practitioner's approach,Mc Graw Hill