ON THE FELICITOUS APPLICATIONS OF NATURAL LANGUAGE

Muhammad Shumail Naveed, Muhammad Sarim, Kamran Ahsan
Department of Computer Science, Federal Urdu University of Arts, Science & Technology, Karachi, Pakistan
shumail.naveed@fuuast.edu.pk, msarim@fuuast.edu.pk, kamran.ahsan@fuuast.edu.pk

ABSTRACT: *Among all languages, the natural languages are the most powerful, proper and logical way of communication. Natural languages are successfully used in end-user programming, databases, robot controlling and question answering systems. In this paper we have identified the areas in which the use of natural languages can be exercised and helps in achieving their prime objectives. Unifying natural languages with decidable computable problems requires different degrees of efforts. Employing the restricted but expressive subset of the natural language is simple but the use of unrestricted natural language in any application domain is still an open challenge for the researchers.*

Keywords: Natural Language, Natural language programming, Context Free Grammar, Parsers, Semantic Analyzer.

1. INTRODUCTION

Any language that human beings learn from their surroundings and apply to communicate with one another is called natural language [1]. Natural languages are employed to articulate the knowledge, acquaintance and sensations and to communicate our responses to others. In essence the natural languages are the most powerful, proper and logical way of communication. A language is not simply a system of communication, but also a form of power [2].

Artificial languages are created by humans to communicate with their technologies. The term artificial language implies a language specially crafted by humans [1]. Most of the artificial languages are developed to communicate with technologies like computers. All programming languages are artificial languages. Programming languages are developed by humans for expressing algorithms in computational way and instructing the machines.

Scientific study of languages is called linguistics. The detailed studies of languages from linguistics point of view signify that among all the communicational systems, natural languages are the most powerful, effective and precise way of communication, consequently it is viable and attainable to use natural languages in other computational areas.

This paper illustrates some areas in which the natural language may be maneuvered with other computable problems for achieving their sole objectives more adequately. The paper is organized as follows. The second section of this paper provides a concise overview of computing systems in which the natural language is already being applied. The third section is the core of paper and provides the areas in which natural language can be practiced. The forth section describes the conclusion and followed by the future work.

2. PREVIOUS WORK

The idea of using of natural language in different computable problem is a not a new idea. Since 1960's several notable amalgamations have been introduced.

Natural language programming is a type of programming language which permits the development of programs by using the natural language [3]. The use of natural language in programming appreciably simplify the inherent complexities of programming and allows the large range of users to program without understanding the unusual and fabricated syntax of programming languages.

Pegasus is one a useful natural programming language. Pegasus is capable to understand natural language and generate executable program from the input. Pegasus is multi lingual and available for English and German. The architecture of Pegasus is modeled in the form of brain. In Pegasus, mind, long term memory and short term memory are the central elements of the brain.

Natural Language Computation (NLC) is another natural programming language used for performing operations on matrices. Principally NLC is developed for the manipulation of data store in tables or matrices [4]. Through NLC it is possible for end users to perform basic matrix operations without understanding the deep knowledge of computing. In NLC all the input sentences are imperative and mostly start with imperative verb. An expressive subset of domain specific English is allowed in NLC.

Metafor is a natural programming language based on the notion of writing computer program as telling a story [5]. By using Python, Metafor automatically generates a program structure from the input story. Metafor used a meaningful and demonstrative subset of English. Straightforward depictions of program objects and their attributes create scaffolding code. Metafor is a valuable brainstorming tool. During programming in Metafor, when a programmer types a story, the environment generates and maintains the side-by-side "visualization" of the user's story.

A programming language that uses natural language for defining their commands is called "natural language supplemented programming" [3]. AppleScript and KlarDeutsch are the best examples of natural language supplemented programming. AppleScript is a scripting language and uses natural language, and its programs highly echoed the structure of programming languages. KlarDeutsch is developed to instruct machines through simple sentences of German language.

NaturalJava is an interface that allows the programmer to develop the program in natural language and translate into java program. Sundance, PRISM and TreeFace are the core components of the NaturalJava [6]. Although NaturalJava supports several facilities of Java, but many are still lacking. There is no support of declaring arrays in NaturalJava. In a similar way nested classes are not supported in NaturalJava.

Inform7 is a programming system based on natural language to developed console based fiction tales.

In [7], Vadas and Curran developed a prototype that accepts unrestricted English and transforms into the equivalent Python programming language code.

A natural language interface to a database (NLIDB) is one of a system that authorizes the user to use natural language for retrieving the data from the database [8]. IRUS, BBN's

PARLANCE, IBM's LANGUAGEACCESS, LOQUI AND ENGLISH WIZARD are the commercial NLIDBs.

SHRDLU is another interface based on natural language and capable to verify commands and make conversations about a world consisting of blocks on a table [7]. SHRDLU can answer the questions typed by the user.

Question answering systems are the specialized tools used for searching and retrieving the information from the textual document. Questions answering systems are successfully augmented with natural language. AnswerBus is one a widely known question answering system which questions in natural language (Portuguese, English, German, Portuguese, French, Italian and Spanish) and reply in English [9]. In [10] the Bengali language based answering systems (BFQA) for factoid questions is introduced and the initial results of its evaluation are very encouraging. Other popular question answering systems are Answerbag, START, Google Knowledge Graph, Blurtit and AllExperts [11].

SNAP (A Stylized NAtural Procedural language) is a natural language based procedural language for nonscientists [12].

HANDS (Human-centered Advances for the Novice Development of Software) is a system developed for children [13], and facilitates the construction of animations and simulations which are interactive in nature.

3. APPLICATION DOMAIN

Due to its innate and intrinsic robustness, natural languages can be used in several problem areas. Following is a concise, albeit effectual discussion about the areas in which natural language can be utilized.

3.1 Linear programming

Linear programming (LP) is one of a significant scientific development of mid-20th century [14]. LP generally deals with assigning limited resources to competing activities in an optimal way. In LP all the mathematical functions are linear functions.

Conventionally the LP problem is solved by analyzing the problem statement and the development of LP model; this process is mostly performed manually. A natural language based system can be developed that performs this process automatically. Architecture of that system would be pretty straightforward. System receives the problem statement (question). The front-end of the system would be comprises of scanner, parser and semantic analyzer. The front-end tokenizes the problem statement, verifies its structure and identifies the decision variables, objective function, objective function coefficients, constraints coefficients and nonnegative constraints. Although there are tens of parsers that might be used in this system, however combinatory categorical grammar (CCG) parser is more appropriate for this system. CCG grammar is a type of categorial grammar in which the function of structures rules is exclusively based on the category of their inputs [15]. Semantic analyzer of the front-end verifies the connotation and identifies the attributes of identifies the decision variables, objective function, objective function coefficients, constraints coefficients and nonnegative constraints. Semantic analyzer also converts the parse tree generated by the CGG parser into discourse representation structures (DRS) predicates, this may be accomplished by using ccg2sem (Prolog package) system.

The back-end of the system receives the input from the front-end and performs mapping to convert DRS predicates into LP model.

3.2 Game theory

Game theory is a mathematical model and area of study deals with decision making. Game theory is generally used when two intelligent opponents with conflicting objectives are striving to surpass other [16].

In the domain of game theory, the terminology "players" is used to represent two antagonists (opponents). Each player in a game has number of strategies or alternatives. Payoff is related with each pair of strategies which is received by one player from other. Such games are called two-person zero-sum games. These games are frequently delineated by payoff matrix. Such matrix is predominantly constructed by solvers manually. However it is possible to construct the payoff matrix automatically by means of natural language supplemented system. Like the natural language augmented system for LP, here the system is also comprises of front-end and back-end. The front-end tokenizes the problem, verifies its structure and denotation. During front-end analysis the lexemes and their possible denotations are discerned through morphological and lexical analysis. The parser identifies the sentence by deciphering the lexicalized text into a tree. CCG or Tree-adjoining grammar (TAG) based parser are more reasonable for this system. TAG is tree-rewriting systems [17]. In TAG the structural attributes of words are encoded as tree structured-objects of extensive volume. The languages generated by TAGs are called tree-adjoining languages, and these languages formalize some strictly context-sensitive languages and included in the class of the mildy context-sensitive language. Indexed languages properly contained the tree-adjoining languages and tree-adjoining languages contained the context free languages [18]. The back-end of the system receives the input from the front-end and performs mapping by generating the payoff matrix.

3.3 Learners programming language

Technicality and hardness are the inherent features of programming. Programming essentially entails the number of specialized dexterities and acquaintance of the structure (syntax) of particular programming language being used. It is very tedious, cumbersome and frustrating for beginners to grasp the grammatical structure of an artificial programming language in that they are obliged to grasp and comprehend syntactic and general programming concepts concurrently. Even professional programmers may be hindered while adapting the syntax of new programming language. Novice programmers face many problems while using conventional programming languages. Contemporary programming languages have myriad many technical details that are not manifested in natural languages. The use of restricted natural language can be used in the design of an educational programming language called the learners programming language. Learners programming language (LPL) is a pre-programming programming language which uses very restricted natural language for writing computer programming in computational style. Learners programming language can be intervened before the first programming course and provide the introductory knowledge to novice students and ultimately help in understanding the actual programming course. LPL should allow the students to write programming in a simple, concise and plain English language, and generates the equivalent high level codes of contemporary programming language from the input program. Learners programming language should be the general purpose programming language and includes the features like: variables, data types, console input and outputs, comments, expressions, conditional structure, iteration structure, one dimension array, user define functions and basic level disk I/O. LPL should help the beginners in learning the concepts of programming languages without engaging them in the tedious syntax of fabricated programming languages. LPL also helps the beginners in learning and adapting contemporary programming languages. The translator for LPL is highly motivated from the structure of compiler and high level translators. Translator consisted of five main phases. The lexical analysis is the first phase which reads the source program, identifies the lexemes and generates the equivalent tokens. Finite state automata and regular expression are the landmark concepts that are used in the development of lexical analyzer. The syntax analysis is second phase of translator for LPL which receive the program in the form of lexemes and tokens from the lexical analyzer. In spite of using restricted natural language, the LPL would be too tiny, and its major portion can be constructed by the context free grammar (CFG). The parser should generate the parse tree that will be received by the semantic analyzer which is the third component of the translator. Semantic analyzer may use the attribute grammar [19] to verify the connotation of the program and update the parse tree. During fourth phase the intermediate code translator convert the program in a form that is highly amenable for high level translation.

The last phase of the translator is high level translation. During this phase the representation of the program instructions written in natural language shall be converted in high level language. In order to realize this task, the notion of programmatic semantic [20-21] would be used.

3.4 CASE NOTE ANALYZER

The profession of law is obviously very important and extremely affects the overall system of any nation or country. Writing legal case notes is a common activity performed in this domain. Writing, reading and analyzing the case notes/reports is indispensable in the study of law and require high level of creativity, analytical skills, logical reasoning, perseverance, reading and writing skills. Mostly these tasks are performed manually, but it is still possible to develop a restricted case specific application, augmented with natural language that systematically performs this task with the proper involvement of technical person(s) through the dialog box. The function and architecture of case note analyzer are moderately tortuous and circuitous. From user point of view, the system should analyze the case, and identifies the material facts, decisional history, area of law, identifies the parties, kind of court, judges, holding, ratio per judge, ratio decidendi and the expected decisions. The generic architecture of the system almost follows the previously defined structure of linear programming system. Through morphological segmentation, the individual words are identified into morphemes and categorized into relevant class. For the given case the part-of-speech tagging is performed. Parser generates the parse tree(s). Probabilistic context free grammar and probabilistic parsing are more adequate for this system. Through relation extraction the association between entities in the note is analyzed. Through topic segmentation the most of the above facets are identified. In case of any ambiguity, system may ask the question from the user. Finally the summary of the case note will be provided to the user.

3.5 Other areas

There are many other areas in which the use of natural language makes the system more viable and serviceable. Scripting programming languages, numerical computations, symbolic computations, home-appliance interfacing, mobile based health-care systems, type-setting documents, socket programming, expert systems and simulation modeling are the few areas in which the natural language can be used and helps in achieving the actual objectives.

4. CONCLUSION

Providing communication between humans is the intrinsic goal of natural language. But due to its solidity and intrinsic wholesomeness, natural languages can be used in several areas. In this paper some of these areas are identified and the notions required for their implementation are discerned at a very high level of abstractions. The use of restricted subset of a natural language is somewhat simple or at least efficiently manageable; however the use of unrestricted natural language in any computational problem is absolutely very difficult and ultimately requires more powerful techniques for the handling of ambiguity, parsing, denotation and translation.

REFERENCES

1. Harris, Marry Dee, *Introduction to Natural Language Processing*, Reston Publishing Company, Inc. A Prentice-Hall Reston. Virginia (1985)
2. Bourdieu, Pierre, *Language and Symbolic Power*, Cambridge Polity Press (1992)
3. Knöll, Roman and Mezini, Mira, "Pegasus: first steps toward a naturalistic programming language", *OOPSLA '06 Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, 542-559(2006)
4. Ballard, Bruce W. and Biermann, Alan W., "Programming in natural language: "NLC" as a prototype", *ACM '79 Proceedings of the annual conference*, 228-237(1979)
5. Liu, Hugo and Lieberman, Henry, "Metafor: Visualizing Stories as Code", *Proceedings of the 10th international conference on Intelligent user interfaces*, 305-307(2005)
6. Price, David and Riloff, Ellen and Zachary, Joseph, and Harvey, Brandon, "NaturalJava: A Natural Language Interface for Programming in Java", *Proceedings of the 5th international conference on Intelligent user interfaces*, 207-211(2000)
7. Vadas, David and Curran. James R., "Programming With Unrestricted Natural Language", *Proceedings of the Australasian Language Technology Workshop 2005*, 191-99(2005)
8. Androutsopoulos, I. and Ritchie, G. D. and Thanisch, P., "Natural Language Interfaces to Databases – An Introduction", *Natural Language Engineering*, 1(1): 29-81(1995)
9. Zheng, Zhiping, "AnswerBus Question Answering System", *In Proceedings of the Second International Conference on Human Language Technology Research*, 399-404(2002)
10. Banerjee, Somnath and Naskar, Sudip Kumar and Bandyopadhyay, Sivaji, "Bfqa: A bengali factoid question

answering system", *In Text, Speech and Dialogue*, volume 8655 of Lecture Notes in Computer Science, 217-224(2014)

11. Lebedeva, Olga, and Zaitseva, Larissa, "Question Answering Systems in Education and their Classifications", *In Joint International Conference on Engineering Education & International Conference on Information Technology*, 359-366(2014)
12. Barnett, Michael P. and Ruhsam, William M., "SNAP: An Experiment in Natural Language Programming", *In Proceedings of the Spring Joint Computer Conference*, 75-87(1969)
13. Pane, John F. and Myers, Brad A. and Miller, Leah B., "Using HCI techniques to design a more usable programming system", *Proceedings of IEEE Symposia on Human Centric Computing Languages and Environments*, 198-206(2002)
14. Hillier, Frederick S. and Lieberman, Gerald J., *Advance Praise For Introduction To Operations Research* (7th Edition), McGraw-Hill Higher Education (2001)
15. Steedman, Mark and Baldridge Jason, "Combinatory categorial grammar", *Non-Transformational Syntax Oxford: Blackwell*, 181-224(2011)
16. Taha, Hamdy A., *Operations Research: An Introduction (8th* Edition), Prentice Hall (2006)
17. Joshi, Aravind K. and Levy, Leon S. and Takahashi, Masako, "Tree adjunct grammars", *Journal of Computer and System Sciences*, 10(1):136-163(1995)
18. Joshi, Aravind K. and Schabes, Yves, "Tree-Adjoining Grammars", *Handbook of Formal Languages*, 69-123(1995)
19. Knuth, Donald E, "The genesis of attribute grammars", *In Attribute Grammars and Their Applications*, Springer Berlin Heidelberg, 1-12(1990)
20. Liu, Hugo and Lieberman, Henry, "Toward a programmatic semantics of natural language", *IEEE Symposium on Visual Languages and Human Centric Computing*, 281-282(2004)
21. Liu, Hugo and Lieberman, Henry, "Programmatic semantics for natural language interfaces", *In CHI'05 Extended Abstracts on Human Factors in Computing Systems*, 1597-1600(2005)