Markov Logics Based Automated Business Requirements Analysis

Imran S. Bajwa, *Member IACSIT*

Abstract — **Automated Software engineering has been an area of interest for NLP scientists for last many decades. Various scientists have investigated applicability of the natural language based interfacing for business/software requirement analysis. Some of them are using rule based approach and some of them have used neural networks and case based reasoning, etc. These techniques provide results up to 70 – 80%. The key objective of this research paper is to investigate the use of recently introduced approach of Markov Logics by Pedro Domingos and have a comparison of the output of this approach with other already employed approaches in the recent past. Results of analyzing the natural language text using Markov Logic has been presented in the next paper.**

Index Terms — **Requirement Specifications, Information Extraction, Business Requirement Analysis, Information Reuse, Text Processing**

I. Introduction

Analysis of system requirements is a major phase of system engineering. In this phase of requirement analysis, major focus is typically to discover the key requirement issues initially and then analyze them individually. Appropriate definition of these requirements and their requisite documentation is the premier concern of this phase. Attuned description of the requirements ultimately holds up in clear and precise definition of the scope of the targeted project. Ultimately this process ends up with the literal estimation of the constraints i.e. time, budget, and resources needed to complete the project. The precise definition of requirements at the early stage facilitates in creating the exactly matching project or product according to the business needs [1]. Recently, the object oriented representation of the business processes and functions was proved to be a major trend in the modern business and software design skews. This trend has also pretentiously affected the exercise of existing tools and techniques in the software engineering methodologies and led up to launch the new tools and techniques of software requirement modeling [2].

Unified Modeling Language (UML) is the major tool for object oriented design of software requirement analysis and modeling. UML supports visual representation of business processes and functions at the multiple dimensions and levels of details and document software assets [3]. On the other hand, the invasion of CASE (Component Added Software Engineering) tools to support computer added object oriented analysis and design. CASE tools i.e. Rational Rose, Smart Draw, Visual UML, GD Pro, MS Visio, etc provide services to draw UML diagrams but these tools have following shortcomings [4]:

- Lack of consistency in documents created by object oriented CASE tools.
- Deficient CASE tools make available a limited range of utilities that are used for applicability of UML standards.
- Tedious nature of user interface of CASE tools stipulate extended efforts from the software analyst.

Various scientists have investigated applicability of the natural language based interfacing for business/software requirement analysis. They have used neural networks, rule based approach, case based reasoning, etc. These techniques provide results up to 70 – 80% [5]. The key objective of this research paper is to investigate the use of recently introduced approach of Markov Logics by Pedro Domingos and have a comparison of the output of this approach with other already employed approaches in the recent past. According to P. Domingos, Markov Logic is an extension of First Order Logic (FOL). FOL proposes use of variables, constants, functions and predicates [6]. Using FOL, knowledge is typically represented by junctions, disjunction and negations of these variables, constants and predicates. In FOL, the predicates and some time functions are major constituents for knowledge representation. By definition, FOL is in censorious to manage more than one dimensions of the knowledge. Typically, the basic constituents of natural languages i.e. phrases, axioms, idioms, etc portray more than one meaning. Using Markov Logic, an additional ally of weights with predicates was added in FOL to cover this conspicuous short fall []. This noteworthy addition to FOL converted this handicap approach to more than functional for knowledge representation of natural languages.

The division of this document is that the methods of analyzing the natural language text using Markov Logic has been presented in the next section. Then a methodology has been presented to generate the software modeling and analysis documents. Then implementation details have been provided with the analysis of the performed experiments. Next to experiments, results and analysis section has been presented with the comparison of Markov Logics and its pros and cons. Section of related work/ literature review has

This research work was conducted in the department of Computer Science & IT, The Islamia University of Bahawalpur and supported in part by the Higher Education Commission, Pakistan. This paper is the continuation of the series of publications under the research project of Natural Language Processing based Software Designing by the Information Processing Research Group (IPRG).

I. S. B. Author is Assistant Professor in the Department of Computer Science & IT, The Islamia University of Bahawalpur. He is regular member IACSIT. (phone: +92 (062)9255466; e-mail: imran.sarwar@iub.edu.pk)

been given at the end to depict overview of the previously presented theories and designed systems for NL text based OO analysis and modeling.

II. Materials And Methods

A. Literature Review

In this section a brief history and survey of research work conducted in the area of natural language processing. Later in this section, the tools and approaches used for providing automated tools for software and business requirement analysis and modeling have been presented.

Information extraction and processing have been a key issue of research in last few decades. This research area primarily lies in the area of natural language processing. Two decades ago, R. J. Abbot suggested that the state of a class or an object can be identified by nouns and the behaviour or functionality of a class or object can be identified by verbs [8] in a sentence. Afterwards, H. Buchholz proposed that nouns not only specify classes and objects but also properties [9] of an object or a class. Later on, S. Naduri proposed that 'associations' in different objects can be pointed out by verbs [10]. Further detailed categorization of associations was done by N. Juristo into binary association, identification association, n-ary association, etc [11]. Hector also presented a semi-natural language (4WL) [12] in 2002 to automatically generate object models from natural language text with a prototype tool GOOAL. K. Li also presented hi his work to solve problems related to NL that can be addressed in OOA [13]. Nan Zhou also proposed another conceptual modeling system based on linguistic patterns [14]. All these methods and approaches are not automatic as they involve system analyst to take many decisions during OO analysis and modeling. On the other hand, these methods were just dealing with only basic OO concepts.

For last few years concept of NL based software analysis and design is getting stronger. Different researchers proposed primary level of frameworks and systems i.e. OOAL [12], UML-Rebuilder [15], LOLITA [16], CM-Builder [17], MOVA [19] etc. No one from these tools is able to extract the complete information i.e. classes, objects and their respective, attributes, methods and associations. Some tools just extract names of classes and some of them just deal with objects. A. Oliveira used natural language constituents for OO data modeling and presented a CASE tool named REBUILDER UML [14]. This tool integrates a module for translation of natural language text into an UML class diagram.

This module uses an approach based on Case-Based Reasoning and Natural Language Processing. But, this case tool needs continuous up gradation of case-base and only deals with class diagrams. LOLITA [15] is another tool to generate an object model from NL text. LOLITA only identifies objects from NL text but it cannot distinguish between classes, attributes and their respective attributes. There was another significant contribution by Harmain and Gaizauskas as they presented another NL based CASE tool named CM-Builder [16]. This CASE tool was restricted to create a primary class model. There was no appropriate mechanism for confining objects from NL text. Borstler and Overmyer presented another NL based system to generate class diagrams from user requirements given in NL text [17], [18]. MOVA [19] is another tool that models, measures and validates the UML class diagrams.

B. Markov Logic Based Text Processing

This section contains the description of the working of the presented algorithm that is based on Markov Logic approach. Markov Logic works on following key issues:

- A logical knowledge base is a set of hard constraints on the set of possible interpretations.
- Convert them to soft constraints as when a predicate or a formula is violated, it becomes less probable not impossible.
- Violation of a formula in knowledge base never means of denial of all possible meanings.
- In place of refuting a possibility of any formula at all, it should be given a percentage or a weight. Here acceptance does not mean 100% approval and denial doesn't mean 100% rejection.

Higher weight → Stronger impact of the constraint

Lesser weight → Weaker impact of the constraint

All above mentioned points are used to write algorithm to analyze NL text and then extract various OO modeling elements. The major steps of the Markov Logic based Text Processing are as following.

The linguistic analysis procedure is initiated with getting input string and string is tokenized into symbols or tokens. Afterwards, tokens of the given text go through POS Tagging to identify different parts of speech. This phase is called morphological analysis. An example of POS tagged sentence is given below. Following figure shows the procedure of POS tagging applied on a piece of text. The processing of the Markov logic based algorithm is shown graphically in the figure 1.0.

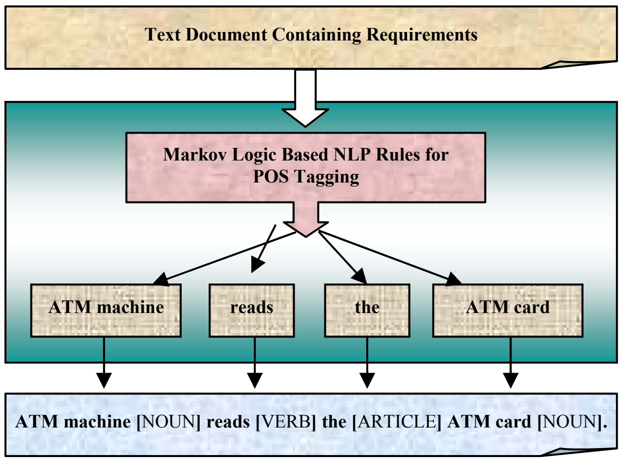
Text Document Containing Requirements
Markov Logic Based NLP Rules for POS Tagging
ATM machine
reads
the
ATM card
ATM machine [NOUN] reads [VERB] the [ARTICLE] ATM card [NOUN].

Fig 1.0- Applying Parts of Speech Tagging on Text

Furthermore, the POS tagged text is lexically analyzed with the help of a parse tree. Syntactic Analysis carried out to validate phrases and sentence according to grammatical rules defined by the English language. This step also helps in identifying the main parts of a sentence; object, subject, actions, attributes, etc.

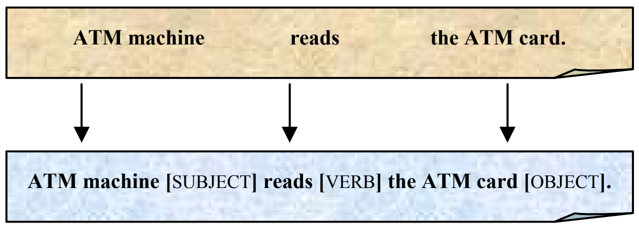

Fig 2.0- Applying Parts of Speech Tagging on Text

After linguistics analysis, classes, objects and their respective attributes and methods will be received. Next step is to find out associations among objects doing semantic analysis. Prepositions are used to identify relation among the actors and their corresponding actions. A logical model of the use case, sequence and activity diagrams is generated as an output on the basis of previously extracted information.

A top down approach is followed in creating a logical model of diagrams from the given text. A drawing module converts the logical model into the use case, sequence and activity diagrams by connecting small pieces of images already stored in database.

III. Text Based Software Requirement Analysis

Business requirement analysis defines on clear basis that what is the need and what is going to be designed. A focused and detailed requirement analysis can ultimately help out a software engineer to get rid of the mistakes and problems possibly occurring at the late stage of the software development. Following steps are typically involved in the text based business/software requirement modeling:

Step 1: Identify Key End-Users

This is the primary phase of software requirement analysis to identify the key people who will be affected by the project. The main focus of this phase is to clarify exactly the related and concerning people to the project and the end user of the project who will use the solution, product, or service. Business/software analyst must carefully collect the inputs/views/needs/requirements of the end users.

Step 2: Capture End-Users Requirements

The exact requirement can be conceived by the end users of the targeted project or service. These are the people who will help the business/software analyst to define the exact scope of the project's or service's functionalities. There are multiple options to collect the requirements i.e. interview, questioners, use cases etc.

Step 3: Requirements Classification

All the requirements collected through any means are gathered at a point. For unified modeling of the requirements, the gathered data is needed to be identified and classified into following categories.

- *Functional Requirements* – These requirements describes the beahviour of the various components or functions also called behavioural requirements. These requirements help out in system design and describe the end-user's way of communication with the system. Functional requirements are represented using Use Case diagrams. If functional diagrams are represented through scenarios then sequence diagrams can also be used for representation.
- *Operational Requirements* – These requirements specify the desired capabilities of a system. Operational feasibility and effectiveness of the desired system is highlighted using operational requirements and they are best captured in activity diagrams.
- *Technical Requirements* – These define the technical issues that must be considered to successfully implement the process or create the product. Context diagrams are typically used to represent technical requirements.
- *Transitional Requirements* – A transition is a progression from one state to another and is typically triggered by an event. Transition diagrams are normally used to represent transitional requirements.

Step 4: Linguistic Analysis

In this research paper, only first two types of requirements i.e. functional and operational are addressed. These requirements are represented in the text files. These text files are processed by the Markov Logic based algorithm stated in previous section. First of all the objects, classes and their respective attributes and methods are identified for each type of requirements. After this, following information for each type of respective requirements is extracted from the given text:

- *Use Case Diagram* – For use case diagram, actor and the action performed by the actor is identified. Actor is normally a subject and its associated verb is the action performed by the actor. Some times there are two or more actors. In this case, besides first actor, other actors will be called co-actors.
- *Sequence Diagram* – For sequence diagram, the objects are identifies first and then the respective methods of each object are recognized. The relation between the methods and the sequence of occurring of the methods is also contemporary to find out to draw the sequence diagram.
- *Activity Diagram* – For activity diagram, an operation is needed to be identifies. The related information is that when an operation/activity initiates and what objects are involved and they perform what function during that activity and where the activity completes.

Step 5: UML Modeling

After extracting all the desired information, the extracted facts are translated into respective UML diagrams. To create diagrams all the UML symbols are provided in the form of small images and these small images are joined with each other to create a complete diagram. After creation of a diagram, that diagram is labeled by the respective flags.

IV. Methdology Evaluation

A software system was designed to implement the proposed algorithm. To test the performance of the designed system sample set of text was taken. Afterwards the results of the designed system were compared with the other existing systems. Following is the examples of activity diagram generated from given piece of input text.

A man walks into the hotel. Find out his language. If his language is Urdu, say him “As-Salam-O-Alikum” and if his language is English him, “Hello”. If the customer has a coat then help the customer to put off his coat....

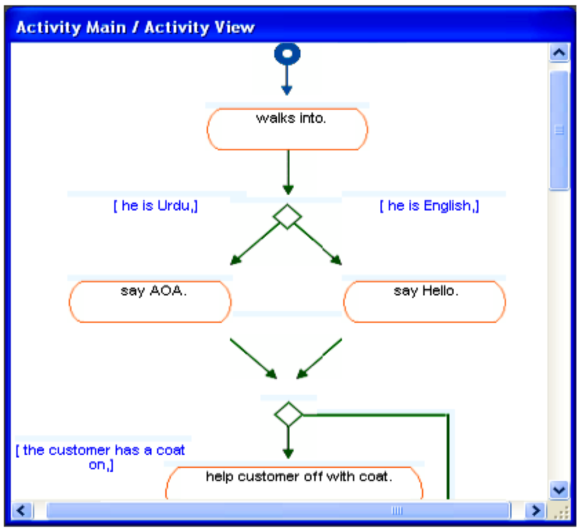
Activity Main / Activity View
walks into.
[he is Urdu,]
[he is English,]
say AOA.
say Hello.
[the customer has a coat on,]
help customer off with coat.

Fig 3.0- Activity diagram generated from input text

Following is the examples of sequence diagram generated from given piece of input text.

Student comes to the teacher and teacher gives him the question paper. Student solves the paper. He submits the paper to the teacher. Teacher marks the paper and submits result to the clerk. Clerk displays the result on the notice board.

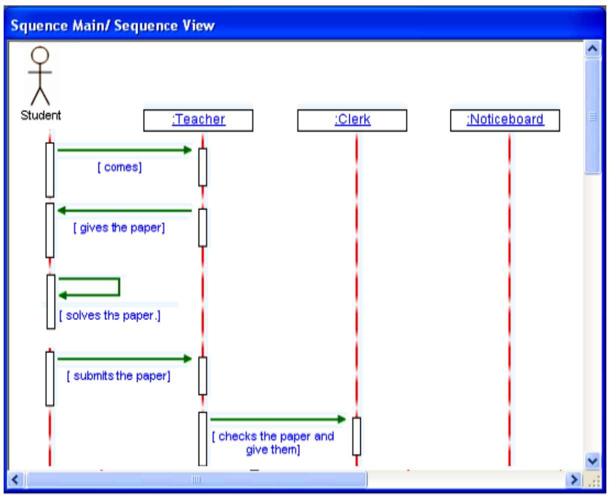

Fig 4.0- Sequence diagram generated from input text

By comparing the results, we can infer the following:

- The designed system has robust capability to identify the actors, use cases, classes and associations among actors and use cases satisfactorily from the given piece of text.
- Markov Logics has been proved a better choice where there is uncertainty or fuzziness among the various constituents of a predicate. Following table shows how the results have been improved with the use of Markov Logic.

To test the accuracy of the diagrams generated by the designed system four parameters were decided i.e. objects and classes, no. of attributes, no. of methods, no. of associations and diagram labeling. Maximum score was declared 25. According to the wrong nominations and extractions, the points were counted. A matrix of results of generated diagrams is shown below.

TABLE I: TESTING RESULTS OF MARKOV LOGIC

Scenario Type	**Objects/ Classes**	**Attribu-tes**	**Methods**	**Relation-ships**	**Label-ing**	**Total**
Simple Rule Based	22	23	21	17	19	82%
Markov Logic	24	25	22	21	21	89%

The above table shows the difference between two approaches i.e. simple rule based approach provides 84% results [7] and Markov Logic based algorithm provides 89% results.

V. CONCLUSION

Markov Logic is a more appropriate solution for natural language processing to generate correct requirements is a difficult task considering the inherent ambiguity in natural language. In this research, the business requirements are analyzed by out designed system. Moreover, these requirements are translated to some UML diagrams by reading and analyzing the given input text in English language provided by the user. The designed system can find out the classes and objects and their attributes and operations using an artificial intelligence technique such as natural language processing. Then the UML diagrams such as Activity dig., Sequence dig., Use Case dig., etc would be drawn.

ACKNOWLEDGMENTS

This work is part of the research project of NLP based Software Modeling undertaken by the Information Processing Research Group (IPRG). I am also thankful to Dr. Irfan Hayder and Dr. M. A. Choudhary for their technical support and guidance during this project work.

REFERENCES

[1] Joseph Schmuller (1991) “Sams Teach Yourself UML”, TechMedia, 1999

[2] Rumbaugh, J., I. Jacobson, and G. Booch, (1998) “The Unified Modeling Language Reference Manual” *Reading, MA: Addison-Wesley*. 1998

[3] Daniel Jurafsky, J.H.M. (2000) “Speech and Language Processing”, *Prentice Hall*

[4] Zahariev, M., [2004] “A linguistic approach to extracting acronym expansions from text”, *KAIS: Knowledge and Information Systems* 6(3), pp. 366-373.

[5] S. Kok and P. Domingos, (2005) “Learning the structure of Markov logic networks’, *In Proc. of ICML-05, Bonn, Germany, 2005. ACM Pres*, pp 441–448

[6] S. Kok, P. Singla, M. Richardson, and P. Domingos, (2005) “The Alchemy system for statistical relational AI”, *Technical report, Department of Computer Science and Engineering*, http://www.-cs.washington.edu/ai/alchemy.

[7] Imran Sarwar Bajwa, M. Abbas Choudhary, (2006) “A Rule Based System for Speech Language Context Understanding” *Journal of Donghua University, (English Edition)* 23 (6), pp. 39-42.

[8] R.J. Abbott, (1983) “Program Design by Informal English Descriptions”, *Communications of the ACM*, Nov. 26(11), pp. 882-894.

[9] H. Buchholz, A. Dusterhoft, B. Thalheim, (1996), “Capturing Information on Behavior with the RADD_NLI: A Linguistic and Knowledge Base Approach”, *Proc. Second Workshop Application of Natural Language to Information System*, IOS Press pp. 185-196

[10] S. Naduri, S. Rugaser (1994), “Requirements Validation via Automated Natural Language Parsing”, *Proc. 28th Hawaii Int’l Conf. Systems Science: Collaboration Tech., Organizational Systems, and Technology*, IEEE Computer Society, pp. 362-367.

[11] N. Juristo and A.M. Moreno, “How to Use Linguistic Instruments for Object-Oriented Analysis”, *IEEE Software*, May/June 2000, pp. 80-89

[12] Hector G. Perez-Gonzalez, (2002) “Automatically Generating Object Models from Natural Language Analysis”, *17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, ACM New York, USA, pp: 86 – 87

[13] K. Li, R.G.Dewar, R.J.Pooley, [2003] “Object-Oriented Analysis Using Natural Language Processing”, *www.macs.hw.ac.uk:8080/techreps/docs/files/HWMACS-TR-0033.pdf*

[14] Nan Zhou, Xiaohua Zhou, (2004) “Automatic Acquisition of Linguistic Patterns for Conceptual Modeling”, *INFO 629: Artificial Intelligence*, Fall 2004

[15] António Oliveira, Nuno Seco and Paulo Gomes (2004) “A CBR Approach to Text to Class Diagram Translation”, *TCBR Workshop at the 8th European Conference on Case-Based Reasoning*, Turkey, September 2006.

[16] L. Mich, R. Garigliano, (1996) "A linguistic approach to the development of object-oriented system using the NL system LOLITA”, *Object Oriented Methodologies and Systems, (ISOOMS)*, LNCS 858, pp. 371-386.

[17] H. M. Harmain and R. Gaizauskas, (2003) “CM-Builder: A Natural Language-based CASE Tool”, *Journal of Automated Software Engineering*, 10, 2003, pp. 157-181

[18] S.L.V. Overmyer, (2001) O. Rambow, "Conceptual Modeling through Linguistics Analysis Using LIDA", *23rd international conference on Software engineering*, July 2001

[19] Manuel Clavel, Marina Egea, Viviane Torres da Silva, (2007) “The MOVA Tool: A Rewriting-Based UML Modeling, Measuring, and Validation Tool”, *in Proc. 12th Conference on Software Engineering and Databases* Zaragoza (Spain), 2007.

[20] J. Borstler, (1996) “User-Centered Requirements Engineering in RECORD - An Overview”, *the Nordic Workshop on Programming Environment Research, Proceedings* NWPER'96, pp. 149-156.

[21] Hugo Liu and Push Singh [2004] “Commonsense reasoning in and over natural language” *Proc. 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems* (KES-2004)

[22] P. Sturt, F. Costa, V. Lombardo, and P. Frasconi, (2003) “Learning first-pass structural attachment preferences using dynamic grammars and recursive neural networks,” *Cognition*, vol. 88, pp. 133–169

[23] Power, R., Scott, D. & Hartley, A. [2003] "Multilingual Generation of Controlled Languages" in *Proc. 4th Controlled Language Applications Workshop* (CLAW03), Dublin, Ireland.

[24] P. C. R. Lane and J. B. Henderson, (2001) “Incremental syntactic parsing of natural language corpora with simple synchrony networks,” *IEEE* Transactions on Knowledge and Data Engineering, vol. 13, no. 2.

[25] Mich, L. (2001) "Ambiguity Identification and Resolution in Software Development: a Linguistic Approach to improve the Quality of Systems" in Proc. Of 17th IEEE Workshop on Empirical Studies of Software Maintenance, Florence, Italy

[26] F. Costa, P. Frasconi, V.Lombardo, and G. Soda, (2003) “Towards incremental parsing of natural language using recursive neural networks,” Applied Intelligence, vol. 19, no. 1–2, pp. 9–25.

[27] J. Henderson, (2003) “Neural network probability estimation for broad coverage parsing,” in Proc. of 10th conference of the European Chapter of the Association for Computational Linguistics (EACL 2003), pp. 131–138.