# Parsing Natural Language Text of Use Case Description

**Prasanth Yalla**[1]**, Nakul Sharma**[2]
[1,2]*Department of Computer Science Engineering, K.L. University, Vijayawada, Andhra Pradesh, India*
[1]*yallaprasanth@gmail.com*, [2]*nakul777@gmail.com*

***Abstract: Software Engineering process has textual description which can be analyzed for generating useful information. In this paper, an attempt is made to parse the natural language text available in form of use case description. The result of parsing text is also presented herein.***

***Keywords: Software Engineering (SE), Natural Language Processing (NLP), Parsing, SDLC, Computational linguistics***

## I. INTRODUCTION

Natural Language Text has always been subject to varied level of understandings and interpretations. It is always been a challenge to understand and develop interpretations from such a text. SDLC has artifacts which aid in completion of a successful software project. As many of these artifacts have textual data, they can be analysed for getting useful information both in regard to the content as well as the quality.

## II. PROBLEM FORMULATION

In this section the basics of SE and NLP with respect to parsing of text is discussed. A generic software process model can be categorised into many types of sub-processes. As the artifacts are result of various sub-processes, they are left to the humans for their interpretation and understanding. There can be several viewpoints on the benefits of automation in software engineering. The humans can think innovatively while the machine may have to use some rule based techniques or neural networks to fulfil the same objective. The natural language artefact is got from analysis phase. The next phase is the design phase of SDLC. The design phase has major artifact as the Unified Modeling Language Diagrams [1]. There are many factors which contribute towards better understanding of concepts in NLP [4]-[5]. Natural Language Text of ontology is generally of following types:-
*1) highly informal*
*2) structural informal*
*3) semi-formal*
*4) rigoursly-formal [2].*

Unified Modelling Language is a means of providing clarity in the textual artifacts by developing diagrams. According to UML 2.0 specification there are nine UML diagrams. Developing ontology can go a long way in understanding and interpretations of any type. Ontology can be used across the rest of phases. But each phase may require a review of how the ontology is changed [4].

Computer software as field can achieve self propagation by automating various phases of software development. Computers as we see today are quite different from what they were in past. There has been as constant evolution in the analysis, design, coding and testing of computer software [12].

### *A. Literature Review*

Jochen in paper testing against Natural Language Text, discusses how test cases are generated using natural language specification as input [5]. This paper also mentions the specific list of steps necessary for making test cases. This paper observes that a software process is needed for NLP based applications [5].

Mark M et. al. discusses the need of developing a coding prototype and also testing methodology in Natural Language Processing (CAC) software. This paper also suggests that an iterative, result driven approach of process model is more appropriate for developing NLP applications [6].

Happel H discusses means of combining Knowledge Engineering (KE) with Software Engineering concepts and methodologies. It also draws parallels between ontology specific phases in Software Engineering. The paper also gives various examples of applications which use Software Engineering [7].

Innab N et.al. have developed a framework and ontology for providing a common notation useful for the benefit of Software Engineers. It also compares the ontology prepared with the Bunge-wand-Weber (BWW) model. The results of comparison (76.66%) are close to BWW model [4].

Mencl M. addresses developing a Pro-case diagram from the behavioural specification. The textual use cases are converted to Pro-cases based on behavioural protocols. Various case studies have been used to check the result of converting textual use cases to Pro-cases [8].

Kai Koskimies et.al. discusses automation in the scenario and state machine diagrams. By using Object Modelling Notation, scenario and state machine diagrams automation tools are developed [9].

978-1-4799-3064-7/14/$31.00©2014 IEEE

Kollar, Thomas et al discuss how a natural language input can be processed by a robot. The paper describes language is mapped onto the structures for robot to understand [10]. Fabian Friedrich et.al. generate a process model by using natural language text. The natural text is scanned for various POS. The paper claims to make 77% of BPMN models accurately by scanning the document for necessary information [12].

This paper addresses the problem of noise present in the Natural Language Text available in use case description. Automation of software is bound to develop following type of noise:-

Noise at sentence level

Noise at paragraph level

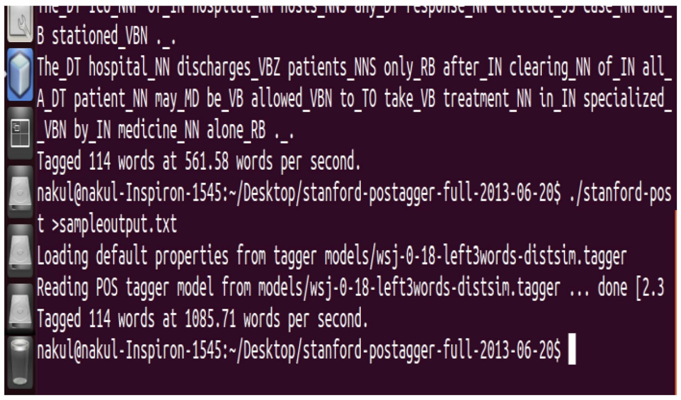
We took an example of parsing Use Case description of Hospital Management System in removing the problem.

## III. PROBLEM SOULTION

To overcome this, an algorithm named NO_REM has been proposed:-

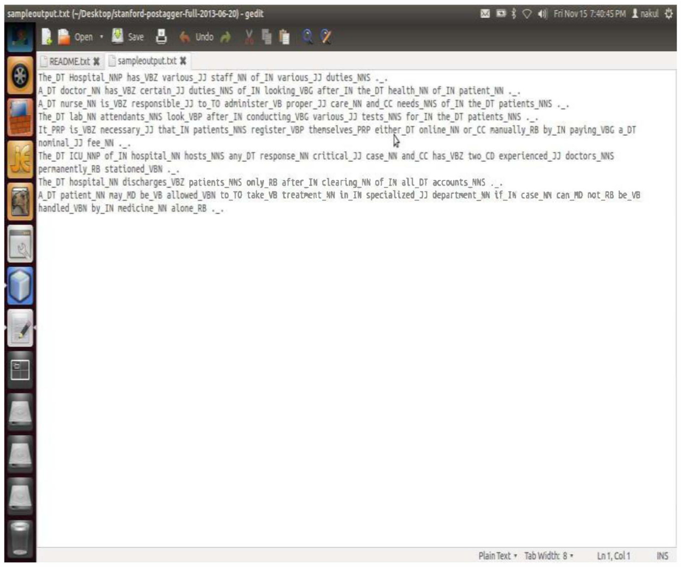
```
NO_REM:

    Input: A POS tagged document

    Output: A noise free document

START

      For all sentences with POS tagged words

      If JJ_occurance >5

            Remove sentence

            If RB _occurance >4

            Remove sentence

            If DT_occurance >4

            Remove sentence
                  STOP
```

The text input should be pre-processed with a POS to identify various Part-of-Speech (POS). There are many tools available for getting the POS information. We had used Stanford's POS tagger [14]. A snapshot of the POS tagged document Fig 2.



```
The_DT ICU_NNP of_IN hospital_NN hosts_NNS any_DT response_NN critical_JJ case_NN and_
B stationed_VBN ._.
The_DT hospital_NN discharges_VBZ patients_NNS only_RB after_IN clearing_NN of_IN all_
A_DT patient_NN may_MD be_VB allowed_VBN to_TO take_VB treatment_NN in_IN specialized_
_VBN by_IN medicine_NN alone_RB ._.
Tagged 114 words at 561.58 words per second.
nakul@nakul-Inspiron-1545:~/Desktop/stanford-postagger-full-2013-06-20$ ./stanford-pos
t >sampleoutput.txt
Loading default properties from tagger models/wsj-0-18-left3words-distsim.tagger
Reading POS tagger model from models/wsj-0-18-left3words-distsim.tagger ... done [2.3
Tagged 114 words at 1085.71 words per second.
nakul@nakul-Inspiron-1545:~/Desktop/stanford-postagger-full-2013-06-20$
```

**Fig.1 Creating POS tagged document**



```
The_DT Hospital_NNP has_VBZ various_JJ staff_NN of_IN various_JJ duties_NNS ._.
A_DT doctor_NN has_VBZ certain_JJ duties_NNS of_IN looking_VBG after_IN the_DT health_NN of_IN patient_NN ._.
A_DT nurse_NN is_VBZ responsible_JJ to_TO administer_VB proper_JJ care_NN and_CC needs_NNS of_IN the_DT patients_NNS ._.
The_DT lab_NN attendants_NNS look_VBP after_IN conducting_VBG various_JJ tests_NNS for_IN the_DT patients_NNS ._.
It_PRP is_VBZ necessary_JJ that_IN patients_NNS register_VBP themselves_PRP either_DT online_NN or_CC manually_RB by_IN paying_VBG a_DT
nominal_JJ fee_NN ._.
The_DT ICU_NNP of_IN hospital_NN hosts_NNS any_DT response_NN critical_JJ case_NN and_CC has_VBZ two_CD experienced_JJ doctors_NNS
permanently_RB stationed_VBN ._.
The_DT hospital_NN discharges_VBZ patients_NNS only_RB after_IN clearing_NN of_IN all_DT accounts_NNS ._.
A_DT patient_NN may_MD be_VB allowed_VBN to_TO take_VB treatment_NN in_IN specialized_JJ department_NN if_IN case_NN can_MD not_RB be_VB
handled_VBN by_IN medicine_NN alone_RB ._.
```

**Fig.2 Sample of POS tagged document**

## IV. CONCLUSION AND FUTURE SCOPE

In this paper we have proposed a use case description can be checked for noise. To prevent spreading of noise, the text is POS tagged and unnecessary sentences are removed. Analysis of natural language text has been going on since quite some time. It remains to be seen how much applications are put in tune with the emerging results. Still a lot can be done in parsing of text in respect to changing requirements.

## REFERENCES

[1] UML Superstructure Specification, v2, Object Management Group. 2004.

[2] Mike Uschold, "Building ontologies: Towards a Unified Methodology", In Proc. Expert Systems' 96, the 16th

Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, 1996.

[3] Sinan Si Alhir, “Learning UML”, O'Reilly & Associates, Third Edition, July 2005.

[4] Nisreen Innab, Ahmad Kayed, and A.S. M. Sajeev, “An Ontology for Software Requirements Modeling”, In Proc. IEEE International Conference on Information Science and Technology, Wuhan, Huber, China, PP-485490.

[5] Jochen L Leidner (2003) “Current Issues in Software Engineering for Natural Language Processing”, In: Proceedings of the Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS) at the Joint Conference for Human Language Technology and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics 2003 (HLT/NAACLâ€™03), pp. 45-50 Edmonton, Alberta, Canada.

[6] "Software Engineering of NLP-based Computer Assisted Coding Applications", Perspectives in Health Information Management, CAC Proceedings; Fall 2006.

[7] Happel, H.-J.; Seedorf, S.: Applications of Ontologies in Software Engineering. In: Proc.of Workshop on Sematic Web Enabled Software Engineering" (SWESE) at ISWC 2006, Athens, Georgia, November 5-9, 2006.

[8] Mencl V.: Deriving Behavior Specifications from Textual Use Cases, in Proceedings of Workshop on Intelligent Technologies for Software Engineering (WITSE04, Sep 21, 2004, part of ASE 2004), Linz, Austria, ISBN 3-85403-180-7, pp. 331-341, Oesterreichische Computer Gesellschaft, September 2004.

[9] Kai Koskimies, Tarja Systa, Jyrki Tuomi, Tatu M, “Automated Support for Modelling OO Software”, IEEE Software, January- February, 1996.

[10] Kollar, Thomas et al. “Toward Understanding Natural Language Directions.” Proceeding of the 5th ACM/IEEE International Conference on Human-robot Interaction - HRI ’10. Osaka, Japan, 2010. 259.

[11] Roger S. Pressman “Software Engineering: A practitioner’s approach”, McGraw-Hill International Edition, 7 Edition. ISBN- 13:978-007-126782-3.

[12] Fabian Friedrich, Jan Mendling, Frank Puhlmann, “Process Model Generation from Natural Language Text”, In Advanced Information Systems Engineering, Eds. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, 482-496.

[13] Jurafsky, D. and Martin, J.H. 2009. “Speech and Language processing. An introduction to natural language processing, computational linguistics, and speech recognition”.

[14] Santorini B, Part-of-Speech Tagging Guidelines for the Penn Treebank Project”, University of Pennsylvania, Department of Computer & Information Science, Technical Reports (CIS).