Integrating Natural Language Processing and Software Engineering

Prasanth Yalla\(^1\) and Nakul Sharma\(^2\)

\(^1\)Professor, K.L. University, Guntur, India
\(^2\)Research Scholar, K.L. University, Guntur, India,
\(^1\)prasanthyalla@gmail.com, \(^2\)nakul777@gmail.com

Abstract

This paper tries to put various ways in which Natural Language Processing (NLP) and Software Engineering (SE) can be seen as inter-disciplinary research areas. We survey the current literature, with the aim of assessing use of Software Engineering and Natural Language Processing tools in the researches undertaken. An assessment of how various phases of SDLC can employ NLP techniques is presented. The paper also provides the justification of the use of text for automating or combining both these areas. A short research direction while undertaking multidisciplinary research is also provided.

Keywords: Natural Language Processing (NLP), Natural Language Understanding (NLU), Software Engineering (SE), Computational linguistics, Software Development Life Cycle (SDLC)

1. Introduction

Software Engineering and Natural Language Processing are related to each other in that both are branches of computer science and engineering. SE is a disciplined approach for construction of a software [1]. NLP is the processing done by computer(s) on natural languages [2]. The paper addresses how both these discipline can be combined and hence increase the chances of universal programmability [11].

Software Engineering consists of tools, methods, processes, techniques for developing software [29]. NLP has as its sub branches various fields which can be utilized in the realm of Software Engineering. It is our conviction by using tools and techniques of one research area in the context of another, better software that will be developed.

There is a lot of research work been carried out in respect of Software Engineering and Natural Language Processing? In our work we are trying to solve the following research question:-

Question- 1. What are the means of combining Natural Language Processing (NLP) and Software Engineering (Software Engineering)?

Question - 2. How can SE and NLP be seen in context of each other?

The paper is divided into following sections as follows. Section 1 gives the brief introduction, section 2 gives the literature review, section 3 provides the analysis of the existing literature, section 4 shows how NLP can be used in Software Engineering context by making use of textual information, section 5 gives the use of SE in NLP software, section 6 gives justification for the use of NL text for automation, section 7 gives the advantages interdisciplinary research section 8 concludes with future scope and direction.
2. Literature Review

Using textual specification, domain model is generated directly. By using NLP tools such as OpenNLP and CoreNLP this work is accomplished. The overall technique involves linguistic analysis and statistical classifiers. Natural Language Text is understood by humans with little effort. The importance of textual processing on natural language text is discussed by Viliam [3].

Farid discusses the use of UML’s class diagram in generation of natural language text. The paper describes various NL based systems to strengthen the view point of generating NL specification from class diagrams. The paper shows use of WordNet to clarify the structure of UML string names and generating the semantically sound sentences [5].

Reynaldo uses controlled NL text of requirements to generate class models. The paper describes some initial results arising out of parsing the text for ambiguity. The paper introduces a research plan of the author to integrate requirement validation with RAVEN project [6].

Deva Kumar, et al., created an automated tool (UMGAR) to generate UML’s analysis and design models from natural language text. They have used Stanford parser, Word Net 2.1 and Java RAP to accomplish this task [7].

Sascha, et al., proposed a round trip engineering process by creating SPIDER tool. The paper addressed the concerns about errors at requirement level being propagated to design and coding stages. The behavioral properties shown from the NL text are utilized to give developer a UML model [8].

Priya More, et al., have developed a from NL text UML Diagrams. They have developed a tool called RAPID for analyzing the requirement specifications. The software used for completing the task is OpenNLP, RAPID Stemming algorithm, WordNet [9].

Waralak, et al., discusses the role of ontology in object oriented software engineering. The author gives the introductory definition of ontology and object modeling. The paper then discusses the development tools and various standards in which ontology can be applied [10].

Walter, et al., suggest the prospect of every human to undertake programming by making universal programmability. The authors predict that by combining NLP, AI and SE, it will be possible to achieve universal programming. The authors are currently developing nlrpBENCH as a benchmark for NLP requirements [11].

Harry M Sneed has undertaken the task of developing test cases from natural language requirements. The NL text is parsed for getting the useful information such as Part-Of-Speech (POS). Using this information, test cases are generated [12].

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 [13].

By using textual business information, UML diagrams are generated by Imran et al., A new methodology for extracting relevant information natural language has been proposed and implemented. The analysis includes information about the amount of objects, attributes, sequence and labeling present with respect to class, activity and sequence diagrams [14].

BrainTool, a tool developed by Riga Technical University, has been utilized in developing UML diagrams from Natural Language Text. A manually generated UML diagrams are compared with the UML diagrams generated from the BrainTool and two-hemisphere technique [15].

Automatic generation of SVBR to UML’s class diagram is conducted with the input specification being put in SVBR format. The main issues in getting UML diagrams from SVBR are presented. Evaluation of NL tools is done using precision and recall [16].
A speech language interface has been developed by using rule based framework. A natural language based automated tool has been used for getting the information objects and their associated attributes and methods [17].

Pro-case diagram from the behavioral specification are developed by Mencl V. The textual use cases are converted to Pro-cases based on behavioral protocols. Various case studies have been used to check the result of converting textual use cases to Pro-cases [18].

How natural language input can be processed by a robot is shown by mapping. The paper describes language is mapped onto the structures for robot to understand [19].

Generating automated scenario and state machine diagrams is shown. By using Object Modeling Notation, scenario and state machine diagrams automation tools are developed [20].

The role of use case diagrams outside the realm of software development is also discussed by Matthias et al. The author suggests role of use case in avionics system and system engineering. The pitfalls of use cases and the solutions are also presented [21].

Arnis, et al., present a meta-model driven approach towards UML’s system as well as simulation. Authors develop the system model by identifying the artifacts from the problem domain and thereby generating Use Case and Activity diagram [22].

Imran S. Bajwa, et al., discusses an approach generating SVBR rules from Natural Language Specification. The paper shows the importance automation in generation SVBR indicating that business analyst with load of documents. They have developed an algorithm for detecting the semantics of English language [23].

Imran S. Bajwa, et al., highlights the cases in which Stanford POS tagger does not identify the particular syntactic ambiguities in English specifications of software constraints. A novel approach to overcome these syntactic ambiguities is provided and better results are presented [24].

Imran S. Bajwa, et al., presents a new model for extracting necessary information from the natural language text. The authors generate Use Case, Activity, Class and Sequence diagram from the natural language text. The designed system also allows generation of system from Natural Language Text [25].

Imran S. Bajwa, et al., propose a SVBR approach to generate a unambiguous representation in English language. The input text is extracted for the relevant information of SVBR. A tool named NL2SVBRviaSBVR is made to accomplish this task [26].

Imran S. Bajwa, et al., propose an interactive tool to draw Use-Case diagrams. The authors have utilized LESSA approach for getting useful information from the Natural Language Text [27].

Mathias, et al., have developed a Requirement Feedback System (REFS) using various NLP tools and techniques. REFS generate UML Models and also checks for the feedback when the requirements are changed [34].

Jochen L. Leidner discusses various issues in Software Engineering for natural language processing. A discussion of toolkit vs framework and system vs experiment is also given [35].

Drigas, et al., have developed a system called Learning Management System (LMS) for the Greek sign language. The system provides the Greek sign language video corresponding to every text [36].

Gang, et al., have resolved several issues in regard to word semantic similarity on web. The author make use of WordNet,’s synonym service to improve the accuracy of word similarity calculator [7].

Yuri, et al., have developed an Internet portal for dissemination computational linguistics knowledge and information resources. The information can be searched according to the subject content or knowledge-based navigation through the portal content [37].
Köhler, et al., propose to integrate UML Diagrams for production control systems. This again, increases the chances of interdisciplinary research [38].

Eladio, et al., propose to utilize state machine diagram in developing program code. The authors have undertaken Systematic Literature Review to accomplish the task [39].

Rogério, et al., have developed a research road map consisting of design space, software engineering processes, from centralized to decentralized control, and practical run-time verification & validation for self-adaptive systems [40].

3. Discussion on Literature Review

There have been notable efforts in generating useful information from Natural Language Text. The information hence generated is used for generation of UML diagrams [5-8, 11-15, 19, 21, 22, 24-27]. In some cases, it has been also been used in generation of process model [13]. Table 1 summarizes some of the contributions made.

<table>
<thead>
<tr>
<th>Paper Title</th>
<th>SE Concept/ Tools</th>
<th>NLP Tool/ Concept</th>
<th>Concept In</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>Domain Model</td>
<td>Standford CoreNLP, Apache OpenNLP with statistical classifiers</td>
<td>SE</td>
</tr>
<tr>
<td>[34]</td>
<td>Eclipse Modelling Toolkit’s (EMF) EMFCompare</td>
<td>Autoannotator, Salmax</td>
<td>NLP and SE</td>
</tr>
<tr>
<td>[8]</td>
<td>UML Model</td>
<td>N L Text</td>
<td>NLP and SE</td>
</tr>
</tbody>
</table>

The analysis of literature hence provides wider coverage to specific use of NLP and SE. The SE has tools, methodologies, and processes etc. which are used in developing the software [29]. NLP also has variety of tools and techniques and its sub branches which can help in developing a more efficient and robust software [2].

The literature review also indicates the scope of combining both the fields is at a lower amount of abstraction which can be increased. Hence in our current work we bring about the necessary information at a higher level of abstraction.

4. NLP in SE

4.1. NLP in Software Development Life Cycle

Software Development Life Cycle (SDLC) consists of set phases which provide guidelines to develop softwares. NLP can be applied to every phase within Software Development Life Cycle [1]. It is specifically more useful when the artifacts of phase or activity are plain text. Plain text can be provided as input to Natural Language Processing tasks. Basically all the activities in which the humans interpret the document there is scope of textual generation [33]. In this section, we try to outline the artifacts in SDLC which fall in the category of plain text. Table 2 shows in analysis phase, which textual documents are, generated [1].

<table>
<thead>
<tr>
<th>Document/ Artifact</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Document</td>
<td>System analyst</td>
</tr>
</tbody>
</table>
The requirement document is authored by the system analyst after understanding the requirements given by stakeholders. Software Requirement Specification (SRS) is a textual written agreement between signed between the company and the stakeholders. Use cases describe the interaction of system to be developed with various actors [1]. Table 3 shows at design level, the textual documents which can be generated [1, 29].

<table>
<thead>
<tr>
<th>Table 3. Design Phase Textual Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Document/Artifact</strong></td>
</tr>
<tr>
<td>Software Design Specification</td>
</tr>
<tr>
<td>UML Diagrams</td>
</tr>
<tr>
<td>Design level Test cases</td>
</tr>
</tbody>
</table>

4.2. NLP in Umbrella Activities

There are the following lists of activities which are performed across various phases in SDLC [1]:-
- Software Project Tracking and control
- Risk management
- Technical Reviews
- Measurement
- Software configuration management
- Reusability management
- Work product preparation and production

The umbrella activities can also have textual artifacts. The artifacts which are read or used by the developers or the business managers will be having textual format. The measurement of cost requires numerical data which is also in numerical format [1]. The exact specification of the any artifact depends upon the organization, team and personal choice of the person executing a particular process [1].

5. SE in NLP

Software engineering although evolving has many standard processes, tools, methodologies which can be utilized in development of NLP software’s. Software development in NLP context can be under following headings:-

5.1. Open Source Development

This type development constitutes giving software for free which facilitates collaborative development of software. In the context of NLP, open source development frees the developer of software from any legal fringes arising from the proprietary licensed software. The software’s many times developed for research purpose and not for commercial development. The researcher’s main focus is hence only to get the prototype ready while patenting the product is left for the industry [35].

5.2. Closed Source Development

Under this model of development, the developed software is given to the customer after payment of a fee. The source code may be made available under certain conditions.
5.3. Software’s Quality Attributes

Software quality strives to create a tangible product that provides an appreciable value to people and develop the software [1]. In NLP systems, the quality is one of the attributes of concern.

There are following quality attributes which need to be assessed for which need to be assessed in NLP software [1]:-

- Performance
- Feature
- Reliability
- Aesthetics
- Perception

6. Justifying the Use of Text

In their paper, Fabia, et al., developed a process model from natural language text [13]. The textual description is quite distinct from the semi-formal or completely formal descriptions. There are following benefits of using any textual documents:-

- It is possible to directly automate the document and artifact.
- The textual information is intelligible to humans.
- Textual format is easy to make.

By having a textual format, it is possible to automate using NLP tools and techniques. The textual artifact can also be converted into any other natural language by undertaking machine translation of the original text. Textual format of the artifact allows a wider audience which can interpret and understand the meaning and revelation behind the subject matter under consideration.

7. Advantages of the Interdisciplinary Research

There are following advantages of undertaking multidisciplinary research:-

1. The different research areas can be combined to get a more holistic view of the common research area. Here for instance, we are trying to see the interdisciplinary research across two research areas, i.e., Software Engineering and Natural Language Processing. By addressing the issues and concerns in both the areas, it is possible to develop a more holistic approach towards Computer Science and Engineering.

2. By undertaking a joint research in both the fields, it will be possible to have greater possibility of automation in the field of Computer Science and Engineering. This is because of automation it is necessary to have textual information or any other type of information which is intelligible to both the computer as well as humans.

3. By having joint research disciplines being developed, it is possible to achieve universal programmability.
Table 5. Our Work and Others Work

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Contribution</th>
<th>Others Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generality of work</td>
<td>More</td>
<td>Less</td>
</tr>
<tr>
<td>Abstraction level</td>
<td>Higher (at subject level)</td>
<td>Lower (at topic level)</td>
</tr>
<tr>
<td>Future Scope and direction</td>
<td>More</td>
<td>Less</td>
</tr>
</tbody>
</table>

Table 5 shows the comparison of the work done which is presented by the authors work and the other similar work. We have tried to mention various parameters under which the comparison can be made. Although standards exist to compare one authors work to another, but still subjectively, we have tried to differentiate out work done with that of the others.

8. Conclusion and Future Scope

In this paper, we tried to develop a vision of combining SE and NLP. The literature review being undertaken is especially in respect of generating UML diagrams from Natural Language Text. The future work entails studying each artifact of SE process models in generating more useful information. The textual specification can be a key in providing the requisite amount of information for carrying out automation. However care needs to be taken to ensure that varying degree of interpretation does not affect the performance of the system. Natural Language processing and Software engineering although divergent, can still be combined with a view of developing a better software.

Acknowledgement

We would like express our thanks and gratitude towards the Head of Department Computer Science and all the staff members of K.L. University who have been a source of inspiration in doing this research work. The second author’s correspondence with Barbara Kichenhem and Dr. Prof. Rajesh Bhatia helped in creating the vision for research work. The second author also thanks to his ME guide, Dr. Prof. Prateek Bhatia for his encouragement and support. Thanks are also due to my mother and father who have helped in every step of life. It is difficult to pen-down the efforts they all have undertaken.

References

[31] “Shell files (sh), Linux file system’s extension”, GNU Licenses.
[34] Dr. P. Yalla uneering, (2013).

Authors

Nakul Sharma, is a PhD Research Scholar in Computer Science and Engineering Department K.L. University. He is pursuing his PhD in the area of Software Engineering. He has completed his M.E. from Thapar University and B. Tech from Bharati Vidyapeeth College of Engineering, Pune. His research interests include Software Engineering and Its Applications.

Prasanth Yalla, received his B. Tech Degree from Acharya Nagarjuna University, Guntur (Dist), India in 2001. M.Tech degree in Computer Science and Engineering from Acharya Nagarjuna University in 2004, and received his Ph.D. degree in CSE titled “A Generic Framework to identify and execute functional test cases for services based on Web Service Description Language” from Acharya Nagarjuna University, Guntur (Dist), India in April 2013. He was an associate professor, with Department of Information Science and Technology in KL University, from 2004 to 2010. Later he worked as Associate professor, with the department of Freshman Engineering from 2011 in KL University. Presently he is working as Professor in the department of Computer Science & Engineering in KL University. Till now he has published 9 papers in various international journals and 4 papers in conferences. His research interests include Software Engineering, Web services and SOA. He taught several subjects like Multimedia technologies, Distributed Systems, Advanced Software Engineering, Object Oriented Analysis and design, C programming, Object-Oriented programming with C++, Operating Systems, Database management systems, UML etc. He is the Life member of CSI and received “Active Participation- Young Member” Award on 13-12-13 from CSI.