Information Retrieval Based Solutions for Software Engineering Tasks Using C Codebases

D. Nishitha[1]*, *S.* Naveena[1], *J.* Vasavi[1], *N.* Trsiha[1], *K.* Charitha[1] and *N.V.* Ganapathi Raju[1]

[1]Information Technology, Gokaraju Rangaraju Institute of Engineering & Technology, Hyderabad

Abstract. Comments are descriptions of laws that a human can understand fluently. It's easier to identify important law blocks with comments. But, not everyone can write the comments duly. They aren't streamlined along with the law. Having outdated comments in the law can affect confusion rather than explanation. This paper aims at removing comments which aren't related to code and not useful using Natural Language Processing(NLP). NLP has come one of the most habituated ways in the analysis of textbooks. In NLP, comments are written from the surrounding code given, and Machine Learning algorithms are applied. A semantic analysis frame for comments using textual and structural features grounded on comment orders and code–comment correlation. A machine learning approach is used to determine whether comments are also consistent and not superfluous based on code and comment correlation.

1 Introduction

The constant exchange of knowledge and experience between researchers and practitioners is crucial for the growth of the field of software engineering(SE).Source-code analysis has extended from its roots in the compiler community to include a range of SE functions. But these roots have left a bias in favour of the types of research a compiler will find valuable. Research without application is pointless, and application without research results in stagnation. Based on this perspective, research activities must be motivated by particular industrial problem statements, and strategies must be evaluated in industrial settings and evaluated by professionals. In recent years, an increasing number of researchers have been extracting data that is useless to compilers. Semantic information discovered in the natural language of a program's source code, such as within the program's identifiers, is a clear example.

Computer programmes can be tagged by adding Human Readable Explanations that describe what the code is doing. When comments are used correctly, they can greatly simplify code maintenance and speed up bug discovery. Comments can serve a variety of purposes. Thus, it is essential to identify whether comments are related to code or not.

* Corresponding author: nishithadusa22@gmail.com

2 Literature Survey

C. Caprile et al. [1] specified about “Analyzing the Language of Function Identifiers", WCRE 1999, pp. 112-122. The Identifiers are the starting point for any program activities that are chosen by programmers which contain all the essential information such as function names, especially when a call chart is available. A regular conceptual classification language is used to analyze the syntactic and semantic structure by the semantation technique of the function identifiers. Through these applications the procedural program database gives the results which have potential use for understanding the program and maintaining such forms of programs.

G. Antoniol, [2] specified about “Linguistic Antipatterns: What They are and How Developers Perceive Them,” specified that Software documentation is a free text and mostly informal in natural language. Some examples are design documents, manual pages, traceability links between source code that require specifications. The programmers need to use meaningful and understandable names for function names, variables, classes and methods. The proposed work includes to recover traceability links from free text documents and source code. A look at mnemonics for identifiers can help to link high-level concepts to program concepts and vice versa, as they often represent the application-domain knowledge programmers process when writing code.

E. Enslen [3] specified "Mining Source Code to Automatically Split Identifiers for Software Analysis. A growing number of automated software engineering tools rely on natural language information gathered from comments and identifiers in the code (e.g. search, interest locations, reuse, quality assessment, etc.). The first step in parsing words from identifiers requires splitting identifiers into their words. Unlike natural languages where the space punctuation is used to separate words, identifiers cannot contain spaces.

H. Asuncion, [4] specified about "Software Traceability with Topic Modeling”. In Software traceability as the projects become complex with the increase in artifacts it becomes one of the most significant tasks in software engineering. The topic modeling is used to automatically record links, this technique combines with machine learning and during software development.

A.Panichella et al. [5] specified that Genetic algorithm-based approaches, information retrieval methods (IR) and especially topic models have recently been used to support essential software engineering tasks (SE) by enabling the search and analysis of software texts.

F. Deissenboeck et al. [6] specified that about 70% of the source code of a software system consists of identifie. Coding style guides somehow deal with this problem, but usually limit themselves to general and difficult-to-enforce rules such as "identifiers should be self-describing". Practically every programming language, however, allows programmers to use almost any string of characters as identifiers, which far too often leads to more or less meaningless or even misleading identifiers.

3 Methodology

3.1 Abbreviations and Acronyms

NLP	Natural Language Processing
ML	Machine Learning
IR	Information Retrieval
SVM	Support Vector Machine

3.2 Obtaining the required data set

A data set of code and comment pairs, along with comment analysis, are provided. A set of 9000 comments (from Github) which contains comment text, surrounding code snippets, and a label that specifies whether the comment is useful or not (a sample shown below).

#	Comment	Code	Label
1	/* uses png_calloc defined in pngriv.h*/	/* uses png_calloc defined in pngriv.h*/ PNG_FUNCTION(png_const_structrp png_ptr) { if (png_ptr == NULL || info_ptr == NULL) return; png_calloc(png_ptr); ...}	Useful
2	/* serial bus is locked before use */	static int bus_reset (. . .) /* serial bus is locked before use*/ { .. update_serial_bus_lock (bus * busR); }	Not Useful
3	// integer variable	int Delete_Vendor; // integer variable	Not Useful

Fig. 3.1. The image contains the dataset containing comments and code with a label (Useful and Not useful).

The development data set contains more than 8,000 rows of comment text, surrounding code snippets, and labels (Useful and Not useful). Date of Release: 1st June 2022.

The test data set will contain 1,000 rows of comment text, surrounding code snippets, and labels (Useful and Not useful). Date of Release: 1st July 2022.

3.3 Equations

The Logistic regression equation can be obtained as

We know the equation for straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + \ldots\ldots + b_nx_n \qquad (1)$$

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}; 0 \text{ for } y = 0 \text{ and } \infty \text{ for } y = 1 \quad (2)$$

The below equation is the final equation for Logistic Regression

$$log[\frac{y}{1-y}] = b_0 + b_1x_1 + b_2x_2 + \ldots\ldots + b_nx_n \quad (3)$$

The Random Forest Classifier equation:

$$\text{rfc} = \text{RandomForestClassifier}() \quad (4)$$

The SVM equation:

$$\text{classifier} = \text{SVC}(\text{kernel='linear', random\_state=0}) \quad (5)$$

3.4 Working

Natural language processing has two main phases: data preprocessing and algorithm development.

Data pre-processing involves preparing and "cleaning" text data for machines to analyze. Pre-processing puts the data into a functional form and highlights elements in the text that the algorithm can work with. This can be done by:

Tokenization:

This is the case when text is broken down into smaller units to work with. Tokenization is about splitting up the data or allowing the algorithm to understand the individual units of the vocabulary. We need to tell the algorithm how to split the data and the words it encounters. Tokenization is therefore the process of dividing strings into a list of tokens.

Stop word removal:

The most common words from the corpus are removed so the unique words tend to give the most meaningful information from the document.

Lemmatization and stemming:

In lemmatization, the morphological analysis of the work is taken into account. This is the case when words are reduced to their root forms in order to process them. This requires detailed dictionaries. Some dictionaries contain all this information. So here it depends on the ending of the word whether it is a kind of conjugated verb or not. Stemming is also different from another technique that is often used in natural language processing, for example. And it's basically turning everything into roots. It is the process of reducing a word to that, to the affixes, the suffixes and prefixes, or the root of a word known as a lima. In other words, it is a technique used to extract the basic form of words by stripping them of their affixes.

Part-of-speech tagging:

NLP is used to analyze texts so that machines can understand how people speak. This is the case when words are marked based on the part of speech they represent - e.g. nouns, verbs and adjectives. This interaction between humans and computers enables real-world applications such as automatic text summarisation, sentiment analysis, topic extraction, named entity recognition, part-of-speech tagging, relationship extraction, inference and more. NLP is widely used for text mining, machine translation and automatic question answering.

3.5 Figures and Tables

Table 1. Logistic Regression classification report.

	Classification Report			
	Precision	**Recall**	**f1-score**	**support**
0.0	0.72	0.72	0.72	358
1.0	0.78	0.78	0.78	447
accuracy			0.75	805
micro avg	0.75	0.75	0.75	805
weighted avg	0.75	0.75	0.75	805

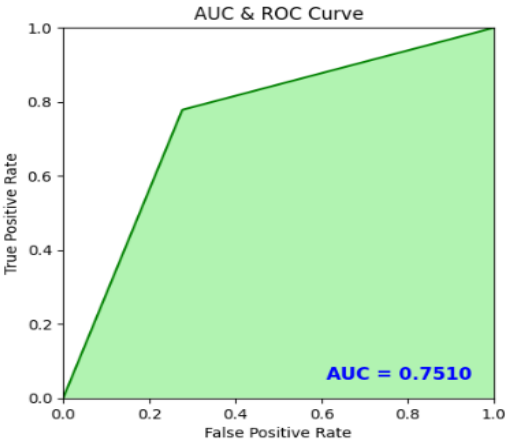

Fig. 3.2. depicts the Logistic Regression model AUC & ROC Curves with an accuracy of 0.7510.

Table 2. Support Vector Machine classification report.

	Classification Report			
	Precision	**Recall**	**f1-score**	**support**
0.0	0.72	0.72	0.72	358
1.0	0.78	0.78	0.78	447
accuracy			0.75	805
micro avg	0.75	0.75	0.75	805
weighted avg	0.75	0.75	0.75	805

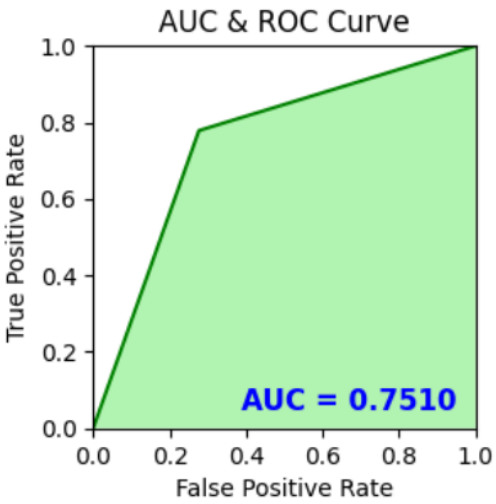

Fig. 3.3. depicts the SVM model AUC & ROC Curves with an accuracy of 0.7510.

Table 3. Random Forest Classifier classification report.

	Classification Report			
	Precision	**Recall**	**f1-score**	**support**
0.0	0.72	0.75	0.74	358
1.0	0.79	0.79	0.79	447
accuracy			0.77	805
micro avg	0.76	0.77	0.77	805
weighted avg	0.77	0.77	0.77	805

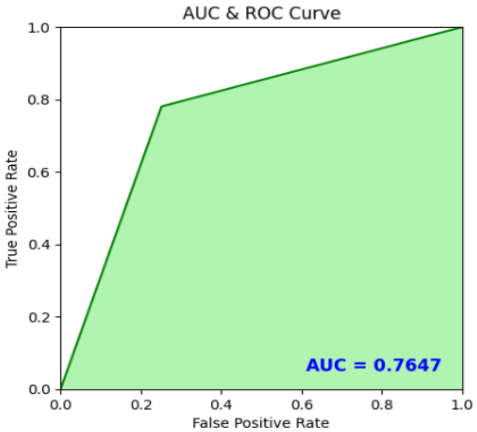

Fig. 3.4. depicts the Random Forest Classifier model AUC &ROC Curves with an accuracy of 0.7647.

4 Result Analysis

As software becomes ubiquitous, the need for tools to support its construction grows. With the help of proper IR techniques combined with natural language processing tools, the model is built to utilize the information contained in the natural language, which helps in the software development process. Since the application of IR to SE is a relatively recent venture, many new applications are likely to emerge. They are likely to take advantage of the diversity of new work in the IR community in the near future. However, as the field matures, more IR-based techniques specifically designed to address SE problems should emerge. The study gives comment classification: A binary classification task to classify source code comments as useful or not useful for a given comment and the associated code pair as input.

5 Conclusion

Comments are typically seen as a good thing. When you add code comments, you can improve readability, clarify what the code is supposed to do, and remember what you need to change or refactor in the future. Software Engineering is becoming so popular these days. So, there are many number of codes present. In order to understand the code, comments will be so useful. But the comments which are not related to the block of code will not be useful. On the other side, they might confuse people who read the code. In order to increase the readability of code, there is a need of removing the comments which are not useful. Here, the model classifies the comments as useful and not useful. By this we can conclude that through our model the useless or meaningless comments in the code is removed in order to increase the readability.

Acknowledgement

We take the immense pleasure in expressing gratitude to our Project supervisor, Dr.N.V.Ganapathi Raju, Professor and Head of the Department, dept of Information Technology, GRIET. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the work. We wish to express our gratitude our Co-coordinators G.Vijendar Reddy, K.Archana for the constant support throughout the research. We express our sincere thanks to Dr. Jandhyala N Murthy, Director, GRIET, and Dr. J. Praveen, Principal, GRIET, for providing us the conductive environment for carrying through our academic schedules with ease. We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

References

1. A. Razzaq, A. Ventresque, The Effect of Feature Characteristics on Location Techniques, in IEEE Transactions on Software Engineering, vol. 48, no. 06, pp. 2066-2085, (2022)
2. C. Caprile, Analyzing the Language of Function Identifiers, WCRE '99: Proceedings of the Sixth Working Conference on Reverse Engineering (1999)

3. F. Deissenboeck, Concise and Consistent Naming, 14(3):261-282 (2006)
4. S. Haiduc and A. Marcus, On the Use of Domain Terms in Source Code, 16th IEEE International Conference, Netherlands, pp. 113-122, .(2008)
5. Lukins, Stacy & Kraft, Bug localization using latent Dirichlet allocation. Information and Software Technology. pp.972-990., (2010)
6. David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning (1st. ed.). Addison-Wesley Longman Publishing Co., Inc., USA.(1989)
7. Arcuri, Andrea & Fraser, Gordon. On Parameter Tuning in Search Based Software Engineering. pp.33-47, (2011)