Automatic comment usefulness judgement via SVM and ANN using contextual token representations

Yogesh Kumar Sahu[1], Ayan Das[2]

[1]*Department of Computer Science and Engineering*
Indian Institute of Technology Dhanbad
[2]*Department of Computer Science and Engineering*
Indian Institute of Technology Dhanbad

Abstract
This paper describes the system submitted by the team from IIT(ISM) Dhanbad in IRSE shared task on automatic judgement of the usefulness of a comment towards an associated source code at FIRE 2022. We have developed a framework where we train a machine learning based model using the neural contextual representations of the comments and corresponding codes to predict whether the comment is relevant to the associated code. In the official evaluation, our system achieves the best F1-score of 0.88 on the test data.

Keywords
Comment-code relevance, Support vector machine, ELMO, Artificial neural network

1. Introduction

For solving any maintenance task, developers spend most of the allocated time reading and understanding the source code before performing any modifications or enhancements. This process is tedious and worsens in the case of unreadable code. The developers often prefer it instead of consulting documents and trackers that are often inconsistent. Reading the comments along with the associated source code can significantly help to apprehend the design of the code and eventually locate the relevant dependencies. The characteristics of comments can be noisy, inconsistent, and sometimes not even be relevant to the source code.

Table 1
Examples of dataset

Comment	Surrounding code context	Label	Explanation
/*READ_INT_FUNCTIONS*/	-5. if (png_ptr != NULL)	Not Useful	The code does not read int,hence the comments is Not Useful
/*Free all memory used in the read struct*/	-10. #ifdef PNG_READ_iTXt_SUPPORTED	Useful	The comment correctly describes the code and hence Useful

Forum for Information Retrieval Evaluation, December 9-13, 2022, India
*Corresponding author
21MT0476@cse.iitism.ac.in (Y. K. Sahu); ayandas@iitism.ac.in (A. Das)

However, they are still easier to follow and hence also, one of the most commonly used documentation approach for software maintenance tasks [1]. Comment analysis approaches have mainly focused on detecting inconsistent comments[2], but the work is somehow less done on the quality and relevance of the information contained in a comment[3]. The comment quality assessment can also help with the do's and don'ts of writing comments thereby can help to develop guidelines for the comment quality assessment. Thus, the approaches to evaluate comments based on whether they increase code comprehensibility for software maintenance tasks are important.

The IRSE shared task of FIRE 2022 focuses on the automatic judgment of the relevance of comments with respect to the corresponding source codes. The task is to determine the usefulness of comments in source codes. It is a binary classification task for determining whether the comment in a given pair of source codes and comment is relevant (useful) to the corresponding source code. The details of the shared task is available in the overview paper [4].

In this paper, we report the description of our system submitted for the task. We have carried out some experiments where we have trained some Machine Learning algorithms that takes the representations of a piece of source code and the corresponding comment as input and predict whether the comment is relevant to the associated source code.

The rest of the paper is organized as follows. In Section 2 we present a review of the related works reported in the literature. In Section 3 we present a discussion of the data made available for the shared task. In Section 4 we present a detailed description of the system submitted for the shared task. In Section 5 we present an analysis of the results of the different experiments. In Section 6 we conclude our work.

2. Related Work

[3]have performed the lexical matching between the code and comment pair to detect redundancy of information.

[5]have conducted a study to derive the attributes of the various categories of task comments used by developers working with Java .They have presented a series of keywords (like todo, fixme) and their likely structure that are used to write comments related to subtasks, short term tasks.

[6] have manually analysed some comments that was from different 3 open source C projects Linux, FreeBSD, and Open Solaris and are randomly sampled for studying their general characteristics and categorise them based on memory, lock and like.

For automated classification and quality evaluation of code comments of C codebases based on how they can help to understand existing code [7, 8] have proposed comment probe based on how they can help to understand existing code. Using neural networks, comments are classified as useful, partially useful, and not useful with precision and recall scores of 86.27% and 86.42%, respectively.

3. Data Description

The training data made available for the shared task contained 8,047 rows of comment text, surrounding code snippets, and labels (Useful and Not useful).Out of the 8,047 rows of comment text and surrounding codes total 3,710 code-comment pair are labelled as Not Useful and 4,337 code-comment pair are labelled as Useful.

The test data provided to us contained 1,001 rows of comment text, surrounding code snippets, and labels (Useful and Not useful).Out of the 1,001 rows of comment text and surrounding codes total 719 code-comment pair are labelled as Not Useful and 282 code-comment pair are labelled as Useful.

Examples of dataset		
Comment	Surrounding code context	Label
/*This should be a binary subdivision search or a hash for*/	-10. png_chunk_error(png_ptr, "Missing PLTE before IDAT");	Useful
/*Finish a chunk started with png_write_chunk_header().*/	-3. png_calculate_crc(png_ptr, data, length);	Not Useful

4. System Description

In this section, we present the details of our system developed for automatic judgement of usefulness of a comment for the associated code.

4.1. Data preprocessing

We have preprocessed the data by defining a function for removing the punctuation marks followed by replacing Not Useful label with 0 and Useful Label with 1. We have extracted out word sequences for comments as well as for codes and after applying above preprocessing we have stored code and comment into a separate lists. We have applied the above preprocessing for both training data as well as for test data.

4.2. Data representation

In the recent NLP systems it is the common practice to represent the data in the form of distributed representations while training different machine learning models. These word representations essentially are neural representation that also contain the information about the contextual words.

We have used ELMO [9] for generating the word representations. The Information Retrieval in Software Engineering(IRSE) Team have provided us with ELMO code link.

ELMO is a type of deep contextualized word representation in which the word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus.

We have extracted the word embedding separately for comments and surrounding code. For each comment we obtained the representations by taking the means of contextual representation to get a resultant sentence embedding of size. For associated code also we followed the same procedure as in for comments.

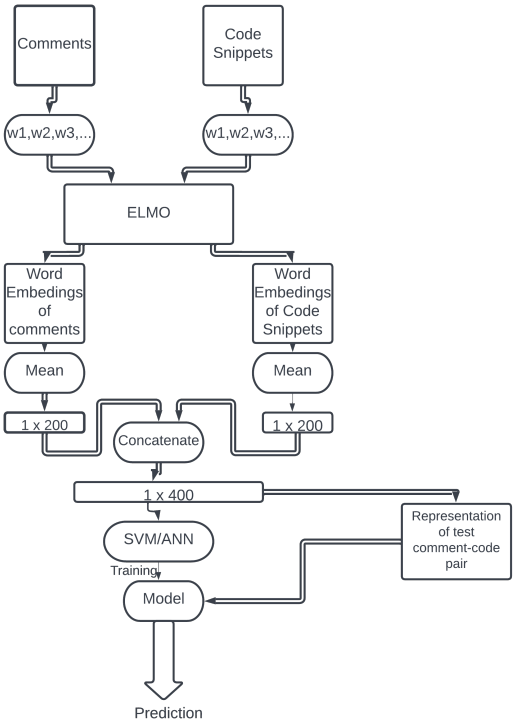

Figure 1: Schematic diagram of our proposed model

For each pair of comments and surrounding code we derived separate representations for the code and comment. Finally we concatenated these two representation to generate a resultant representation for the comment-code pair so obtained.

4.3. System description

We initially trained support vector machine (SVM) model[10] with the linear kernel using the representation so obtained. We also trained SVM model with the Radial Basis Function (RBF) kernel using the representation so obtained. We also experimented and trained Artificial Neural Network[11] using the representation so obtained. The details of the experimental settings are summarized in table 2.

Our artificial neural network based model contains two input layers and we used one output layer. For training the model we have used "Adam optimizer" [12] with a learning rate of 0.0001. In the training process we have taken a batch size of 32 and 50 epochs.

We have applied same preprocessing of data and extracted the representation in the same manner as we did for our training data. We have applied the trained SVM model and Artificial Neural network based models to classify comment text and surrounding code snippets of test data as Useful and Non Useful comments. Followed by we have also applied our trained ANN model to classify comment text and surrounding code snippets of test data as Useful and Non

Table 2
Description of system settings and parameters

Experimental settings		
Sr. No	Model	Settings and parameter used
1.	SVM	Kernel="linear"
2.	SVM	Kernel="RBF"
3	ANN	No. of hidden layers=2 Activation function in hidden layer="relu" Units in 1^{st} hidden layer=200 Units in 2^{nd} input layer=90 Output layer=1 Activation function in output layer="sigmoid" optimizer=Adam learning rate=0.0001 loss function=binary crossentropy batch size=32 epochs=50

Useful comments.

5. Result Analysis

We have applied the trained SVM model(both with linear kernel and RBF kernel) to classify comment text and surrounding code snippets of test data as Useful and Non Useful comments.

5.1. SVM with Linear Kernel

After applying our trained SVM model with kernel as RBF on test data we achieved an overall accuracy of 87.7 %. Confusion matrices are as follows:

		Predicted		
		Useful	Not Useful	Total
Actual	Useful	245	37	282
	Not Useful	86	633	719
	Total	331	670	1001

Precision,recall,F1 score are as follows:

Results			
Label	Precision	Recall	F1-score
Useful	0.74	0.87	0.80
Not useful	0.94	0.88	0.91

When using SVM with linear kernel as our model ,for predicting useful comment-code pair we achieved Precision ,Recall, F1-score as 0.74,0.87,0.80 respectively whereas for predicting Non-useful comment-code pair we achieved Precision, Recall and F1-score as 0.94,0.88 and 0.91 respectively.

5.2. SVM with RBF Kernel

After applying our trained SVM model with kernel as RBF on test data we achieved an overall accuracy of 92.4%. Confusion matrix are as follows:

		Predicted		
		Useful	Not Useful	Total
Actual	Useful	270	12	282
	Not Useful	64	655	719
	Total	334	667	1001

Precision,recall,F1 score are as follows:

Results			
Label	Precision	Recall	F1-score
Useful	0.81	0.96	0.88
Not useful	0.98	0.91	0.95

When using SVM with RBF kernel as our model ,for predicting useful comment-code pair we achieved Precision ,Recall and F1-score as 0.81,0.96 and 0.88 respectively whereas for predicting Non-useful comment-code pair we achieved Precision ,Recall and F1-score as 0.98,0.91 and 0.95 respectively. We noticed a significant increase in Precision recall,F1-score for both predicting Useful as well as Not useful comment-code pair as compared to SVM with linear kernel.

5.3. Artificial Neural Network

After applying our trained ANN model with on test data we achieved an overall accuracy of 91.7 %.

Confusion matrix are as follows:

		Predicted		
		Useful	Not Useful	Total
Actual	Useful	261	21	282
	Not Useful	62	657	719
	Total	323	678	1001

Precision,recall,F1 score are as follows:

Results			
Label	Precision	Recall	F1-score
Useful	0.81	0.93	0.86
Not useful	0.97	0.91	0.94

When using ANN as our model ,for predicting useful comment-code pair we achieved Precision ,Recall and F1-score of 0.81, 0.93 and 0.86 respectively whereas for predicting Non-useful comment-code pair we achieved Precision ,Recall and F1-score as 0.97, 0.91 and 0.94 respectively.

As compared to our previous model (SVM with RBF kernel) the precision,recall and F1-score dropped while using ANN as our Model.

The accuracy of our best model for this shared task turns out to be 92.4%. Best precision for predicting useful comments is 0.81 which is given by ANN and SVM with RBF kernel, whereas precision for predicting Non-useful comments is 0.98 which is given by SVM with RBF kernel.

Best recall for predicting useful comments is 0.96 which is given by SVM with RBF kernel,whereas recall for predicting Non-useful comments is 0.91 which is given by SVM with RBF

kernel and ANN.

Best F1-score for predicting useful comments is 0.88 which is given by SVM with RBF kernel, whereas recall for predicting Non-useful comments is 0.95 which is given by SVM with RBF kernel.

6. Conclusion

In this work we report our system submitted for participating in the IRSE shared task of FIRE 2022. The shared task is aimed towards development of a system that takes a source code comment and the associated source code as input and predicts whether the given comment is useful or not useful for the given source code. We have come up with a machine learning based system that takes the representations of the source code and corresponding comment derived from the distributed contextual representations of the constituent words as input and predicts the usefulness of the comment. To this end, we have trained support vector machine with different kernels and artificial neural network. Our best performing system achieved an best F1-score at 0.88.

References

[1] S. C. B. de Souza, N. Anquetil, K. M. de Oliveira, A study of the documentation essential to software maintenance (2005) 68–75. URL: https://doi.org/10.1145/1085313.1085331. doi:10.1145/1085313.1085331.

[2] L. Tan, D. Yuan, G. Krishna, Y. Zhou, /*icomment: Bugs or bad comments?*/, SIGOPS Oper. Syst. Rev. 41 (2007) 145–158. URL: https://doi.org/10.1145/1323293.1294276. doi:10.1145/1323293.1294276.

[3] I. K. Ratol, M. P. Robillard, Detecting fragile comments (2017) 112–122. doi:10.1109/ASE.2017.8115624.

[4] S. Majumdar, A. Bandyopadhyay, P. P. Das, P. D Clough, S. Chattopadhyay, P. Majumder, Overview of the IRSE track at FIRE 2022: Information Retrieval in Software Engineering, in: Forum for Information Retrieval Evaluation, ACM, 2022.

[5] A. T. T. Ying, J. L. Wright, S. Abrams, Source code that talks: an exploration of eclipse task comments and their implications to repository mining, ACM SIGSOFT Software Engineering Notes (2005).

[6] Y. Padioleau, L. Tan, Y. Zhou, Listening to programmers taxonomies and characteristics of comments in operating system code (2009) 331–341. URL: https://doi.org/10.1109/ICSE.2009.5070533. doi:10.1109/ICSE.2009.5070533.

[7] S. Majumdar, A. Bansal, P. Das, P. Clough, K. Datta, S. Ghosh, Automated evaluation of comments to aid software maintenance, Journal of Software: Evolution and Process 34 (2022). doi:10.1002/smr.2463.

[8] S. Majumdar, S. Papdeja, P. P. Das, S. K. Ghosh, Comment-Mine—A Semantic Search Approach to Program Comprehension from Code Comments, Springer Singapore, Singapore, 2020, pp. 29–42. URL: https://doi.org/10.1007/978-981-15-2930-6_3. doi:10.1007/978-981-15-2930-6_3.

[9] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations (2018) 2227–2237. URL: https://aclanthology.org/N18-1202. doi:10.18653/v1/N18-1202.
[10] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.
[11] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics 5 (1943) 115–133.
[12] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2014). URL: https://arxiv.org/abs/1412.6980. doi:10.48550/ARXIV.1412.6980.