Information Retrieval in Software Engineering utilizing a pre-trained BERT model *

Koyel Ghosh[1], Apurbalal Senapati[1]

[1]*Central Institute of Technology, Kokrajhar, Assam, India*

Abstract

The task is to detect whether a source code comment is useful or not for a given comment, and the surrounding code is paired together as input. IRSE (Information Retrieval in Software Engineering) shared task organized by FIRE 2022 (Forum for Information Retrieval Evaluation), gives a binary classification task where a system classifies *Comments* and *Surrounding Code Context* pairs into two classes: (a) USEFUL or (b) NOT_USEFUL. To do the task, we experimented with the roberta-base model, and the result was 0.9047 in F1 Marco. Our submission gets the second position out of all submissions.

Keywords

Information Retrieval in Software Engineering, BERT, Binary classification

1. Introduction

FIRE 2022 has organized a shared task IRSE[1], where they share some datasets for the Comment Classification task. It is 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. Here, the **Input:** A code comment with corresponding lines of code (written in C) **output:** A label (Useful or Not Useful) in assisting developers in understanding the associated code.

As this is one kind of text classification, word embedding such as wor2vec[2] etc. is needed first. We may use several classifiers like RNN (Recurrent Neural Network) [3], LSTM (Long Short-Term Memory)[4], BiLSTM (Bidirectional Long Short-Term Memory), etc. Here, we use the BERT model roberta-base[5] for this work.

The rest of the paper is structured as follows. Section 2 is the work related to the same. Section 3 describes the experimental setup, i.e., the dataset and a pre-trained BERT model. Section 4 shows the result. Finally, it is concluded in Section 5.

2. Related work

In the paper [6], they have collected 20,206 comments from open-source Github projects and annotated them with assistance from industry experts. Later they use neural networks to

Forum for Information Retrieval Evaluation, December 9-13, 2022, India

*Corresponding author.

ghosh.koyel8@gmail.com (K. Ghosh); a.senapati@cit.ac.in (A. Senapati)
https://github.com/BrainLearns (K. Ghosh)

classify comments as useful, partially useful, and not useful. Their result was precision and recall scores of 86.27% and 86.42%, respectively. As per [7], annotating programs with natural language comments is a standard programming practice to increase the readability of code. They manually annotate concepts for 5600 comments extracted from 672 C/C++ files/projects crawled from code repositories like GitHub. Comment-Mine extracts 38,992 concepts, out of which 79.8% is correct and validated using manual annotation.

3. Experimental Setup

3.1. Dataset

IRSE, a shared task organized by FIRE (Forum for Information Retrieval Evaluation), published the dataset containing 8047 *Comments* and *Surrounding Code Context* pairs training set along with *Class*, i.e. useful or not_useful. A total of 1001 *Comments* and *Surrounding Code Context* pairs are given on the test set. Table 1 shows the details dataset statistics.

Label encoding: Here, we just convert *N*OT_USEFUL to “0” and *U*SEFUL to “1” for the Class column.

IRSE	NOT USEFUL	USEFUL	Total
Training set	3710	4337	8047
Test set	719	282	1001

Table 1
Class distribution analysis for training and test set of IRSE dataset

3.2. Pretrained BERT models

BERT models are trained on a large raw text (without human labeling) corpus in a self-supervised way. Figure 1 shows the representation of the approach. We did several experiments and found the below-mentioned (Table 3) best hyperparameter combination.

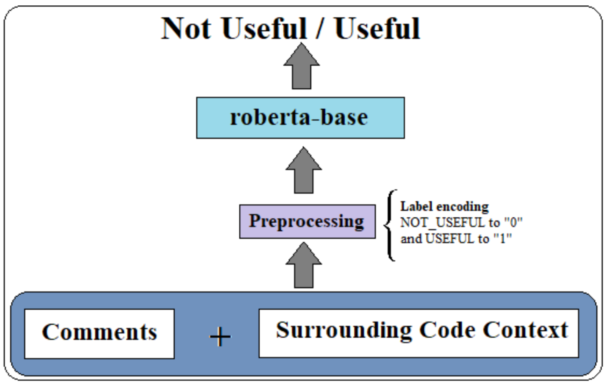

Figure 1: Representation of the architecture to perform the task where roberta-base is used

- **roberta-base[1]:** It is pre-trained on English corpus in a self-supervised manner. It is also case-sensitive.

Hyperparameter	**BERT variants**
Learning-rate	1e-5
Epochs	10
Max seq length	512
Batch size	5

Table 2
Combination of hyperparameters for training roberta-base on IRSE dataset

4. Result

Here, table 3 shows the result, and we put the roberta-base model's result. To evaluate the roberta-base model, we use two class precisions (P_{NOT_USEFUL}, P_{USEFUL}), recalls (R_{NOT_USEFUL}, R_{USEFUL}), F1 scores ($F1_{NOT\_USEFUL}$, $F1_{USEFUL}$) then calculate Macro F1 scores(M_{F1}) here. At last, we calculate *Accuracy*.

$$P_{NOT\_USEFUL} = \frac{True_{NOT\_USEFUL}}{True_{NOT\_USEFUL} + False_{USEFUL}} \tag{1}$$

$$P_{USEFUL} = \frac{True_{USEFUL}}{True_{USEFUL} + False_{USEFUL}} \tag{2}$$

$$R_{NOT\_USEFUL} = \frac{True_{NOT\_USEFUL}}{True_{NOT\_USEFUL} + False_{NOT_{U}SEFUL}} \tag{3}$$

$$R_{USEFUL} = \frac{True_{USEFUL}}{True_{USEFUL} + False_{NOT\_USEFUL}} \tag{4}$$

$$F1_{NOT\_USEFUL} = 2 * \frac{P_{NOT\_USEFUL} * R_{NOT\_USEFUL}}{P_{NOT\_USEFUL} + R_{NOT\_USEFUL}} \tag{5}$$

$$F1_{USEFUL} = 2 * \frac{P_{USEFUL} * R_{USEFUL}}{P_{USEFUL} + R_{USEFUL}} \tag{6}$$

$$M_{F1} = \frac{F1_{NOT\_USEFUL} + F1_{USEFUL}}{2} \tag{7}$$

$$Accuracy = \frac{True_{NOT\_USEFUL} + True_{USEFUL}}{T_{NOT\_USEFUL} + T_{USEFUL}} \tag{8}$$

Where $True_{NOT\_USEFUL}$ = True-negative (model predicted the texts as NOT_USEFUL, and the actual value of the same is also NOT_USEFUL), $True_{USEFUL}$ = True-positive (model predicted the texts as USEFUL, and the actual value of the same is also USEFUL), $False_{NOT\_USEFUL}$ = False-negative (model predicted the texts as NOT_USEFUL, but the true value of the same is USEFUL), $False_{USEFUL}$ = False-positive (model predicted the texts as USEFUL, but the true

[1]https://huggingface.co/roberta-base

value of the same is NOT_USEFUL), P_{NOT_USEFUL} = Precision of NOT_USEFUL class, P_{USEFUL} = Precision of USEFUL class, R_{NOT_USEFUL} = Recall of NOT_USEFUL class, $F1_{NOT\_USEFUL}$ = F1 score of NOT_USEFUL class, $F1_{USEFUL}$ = F1 score of USEFUL class, T_{NOT_USEFUL} = The total number of NOT_USEFUL class text present in the test set, T_{USEFUL} = The total number of USEFUL class text present in the test set.

We execute our code up to 10 epochs and take the best result out of all the epochs. Here, we notice overfitting while fine-tuning pre-trained BERT models. After epoch 4, validation loss increases, and training loss decreases. We didn't try dropout layer here.

Model on	**Precision**		**Recall**		**F1 score**			**Accuracy**
IRSE	**0**	**1**	**0**	**1**	**0**	**1**	**Macro**	
roberta-base	0.9201	0.8678	0.9258	0.9063	0.9229	0.8866	0.9047	0.9178

Table 3
Precision, Recall, F1 score, and Accuracy of roberta-base

5. Conclusion

In this paper, our task is to classify a *comment* and *Surrounding Code Context* pair to *USEFUL* or *NOT_USEFUL*. We used a pre-trained BERT model. During the method, we realize that the maximum length of a *comment* for the entire set is six, and for *Surrounding Code Context*, it's 821. As BERT's maximum input length capacity is 512, we can experiment with longformer[2], but it needs a good configuration machine otherwise may face memory issues. Later, dual BERT[3] can be used in place of a single BERT.

References

[1] S. Majumdar, A. Bandyopadhyay, P. P. Das, P. D Clough, S. Chattopadhyay, P. Majumder, Overview of the IRSE subtrack at FIRE 2022: Information Retreival in Software Engineering, in: Working Notes of FIRE 2022 - Forum for Information Retrieval Evaluation, ACM, 2022.

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Advances in Neural Information Processing Systems 26 (2013).

[3] A. Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, CoRR abs/1808.03314 (2018). URL: http://arxiv.org/abs/1808.03314. arXiv:1808.03314.

[4] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, CoRR abs/1907.11692 (2019). URL: http://arxiv.org/abs/1907.11692. arXiv:1907.11692.

[2]https://huggingface.co/docs/transformers/model_doc/longformer
[3]https://towardsdatascience.com/siamese-and-dual-bert-for-multi-text-classification-c6552d435533

[6] 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.

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