Recent Trends in Intensive Computing
M. Rajesh et al. (Eds.)

doi:10.3233/APC210203

Using Artificial Intelligence in Source Code Summarization: A Review

Shraddha Birari [a,1], Sukhada Bhingarkar [a]
[a] *Computer Science and Engineering, MIT World Peace University*
Pune, India

Abstract. Source code summarization is the methodology of generating the description from the source code. The summary of the source code gives the brief idea of the functionality performed by the source code. Summary of the code is always necessary for software maintenance. Summaries are not only beneficial for software maintenance but also for code categorization and retrieval. Generation of summary in an automated fashion instead of manual intervention can save the time and efforts. Artificial Intelligence is a very popular branch in the field of computer science that demonstrates machine intelligence and covers a wide range of applications. This paper focuses on the use of Artificial Intelligence for source code summarization. Natural Language Processing (NLP) and Machine Learning (ML) are considered to be the subsets of Artificial Intelligence. Thus, this paper presents a critical review of various NLP and ML techniques implemented so far for generating summaries from the source code and points out research challenges in this field.

Keywords. Deep Learning, Neural Networks, Software maintenance, Source code, Source code summarization

1. Introduction

Summarization can be viewed as transformation of data into a concise yet meaningful representation which could be used further for analysis or storage. There are different use cases of the summarization like Document summarization [1][2], Newsletter summarization [3], Summarization of information over social media, Generating summaries over videos [4], Summaries of the content within email [5]and Summaries over the programming language code.

Source code summarization is a methodology of understanding the functionality performed by the source code and automatically generating the descriptions out of the source code. Proper documentation makes the software system maintainable. Correct, complete, and consistent documentation leads to successful maintenance of the software system [6]. The comment written with respect to source code should exactly specify the functionality performed by the code. The comment should include all the necessary information which is performed by the source code. Also, the comment

[1] Shraddha Birari, Computer Science and Engineering, MIT World Peace University, Pune, India
Email: bshraddha30@gmail.com.

written should follow the same format throughout the whole document.

In this paper we present a review of various source code summarization methodologies implemented so far using different natural language and machine learning techniques.

2. Need of Source Code Summarization

This section gives brief idea about the different use cases where source code summaries or comments written are useful.

2.1. Software Maintenance

The software document is an artifact which communicates the information about the software system to the people implicated in the production of that software. The people involved here are the customers, developers, project leaders as well as managers [7]. According to the survey conducted regarding the problems faced by developers in [8], 66 % developers faced challenges while understanding motive or purpose of the piece of code. Understanding code written by someone else is a serious problem rated by 56% developers. 17% developers find it difficult to understand their own code written just a while ago. This shows properly written documentation regarding the source code is important in the software maintenance.

2.2. Code Categorization

In software development finding relevant application in similar category is important to understand the functionality and useful for reusing the common functionality [9]. Programmers can save much time by finding the functionality in similar applications. For finding some application, one can search according to categories such as "Business", "Communication", "Audio/Video", "Games" etc [10]. The problem to find relevant functionality in similar category is due to the mismatch between descriptions written and implementation done. Thus, well written description written in natural language will be helpful in the code categorization.

2.3. Code Search

A code search is general activity while developing a source code. According to the survey conducted among developers regarding the code search [11], 33.5 % people search to refer the example code. 26 % people read or explore the code. 16% search for code localization like "*where class is instantiated*". 16% refer the code to determine the impact like "*understanding the dependencies*" or "*find side effect of proposed changes*".8.5% refer for metadata like "*who recently touched the code*". Also, according to the same survey, on an average developer creates 12 search queries as per each working day. This show searching a code is a frequent and highly important activity. The performance of the code search is highly depending on the text involved in the search term and the code snippets [12]. Code search is difficult when search term

specified as input do not have the same words as the corresponding source code. Thus, well written comments will lead to effective code search.

3. Source Code Summarization Methodologies

As discussed earlier, source code summarization is task which generates the comments from the given code snippet. In this section, we present some existing techniques implemented for source code summarization which includes encoder-decoder model, language model, and reinforcement learning etc.

3.1. Deep reinforcement learning based code summarization.

Reinforcement learning is paradigm in Machine Learning which takes suitable action to maximize the reward for given situation. In [13], for generating relevant summary from the source code, deep reinforcement learning methodology is used. This framework is also known as Actor-Critic network which is used to generate the comments. The confidence to predict the next word is calculated by Actor network whereas Critic network evaluates the value calculated by the actor network. Figure. 1 shows the overview of this framework.

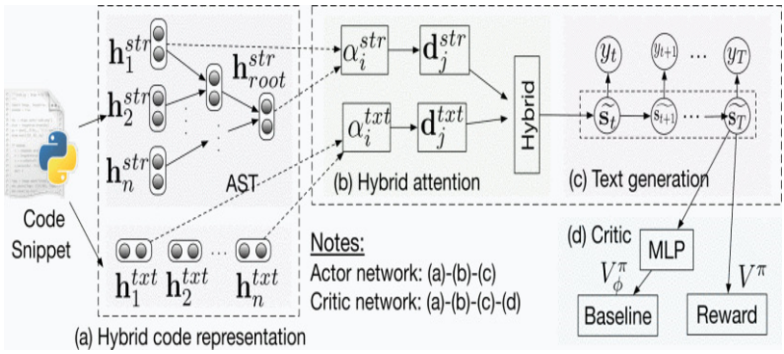

Figure 1. Overview of proposed code summarization framework [13].

This proposed architecture is based on encoder-decoder framework. Hybrid code representation is "Encoder" in this encoder-decoder framework. In this work, sequential tokens as well as structural information is utilized for the code representation. The sequential tokens of the code snippet are denoted by Long Short-Term Network (LSTM) whereas structural information of the source code called as semantics is preserved by Abstract Syntax Tree (AST). AST also maintains the hierarchical representation of the source code. LSTM based on AST represents the structural contents of the code snippet. Thus, sequential as well as structural contents of the code snippet are represented in the "Hybrid code representation".

Actor-critic network acts as a "Decoder" in this encoder-decoder framework. Actor network calculates next word's probability distribution with respect to the current state. Softmax function is used to predict the next word. Critic network approximates the

value generated by the actor network at each time step. In this work, the evaluation score i.e., Bilingual Evaluation Understudy (BLEU) is defined as reward.

The proposed work effectively captures the semantics of the code snippet due to use of AST. Also, actor-critic network predicts the summary, which resolves exposure bias issue. But this model is evaluated only on certain Python code and comment pairs, which may not represent all the types of comments and not generalized for other programming language. Also, only BLEU score is utilized in order to calculate the reward, which may not satisfy human evaluation criteria.

3.2. Statistical Machine Translation based code summarization.

Statistical Machine Translation (SMT) is a methodology for translation among two natural languages, which is adopted in [14]. SMT is basically Natural Language Processing technique (NLP) which identifies grammatical rules from the two languages such as English, Hindi and translates into corresponding language. Phrase Based Machine Translation (PBMT) is utilized which identifies phrase to phrase relationship among source and destination sentence. In PBMT modeling, set of phrase pairs introduced, which include source sentence and target subsequence. Consider an example for the statement: if x %5 == 0, Its phrase pair is: "if" → "if", "x" → "x", "% 5" → "by 5", "== 0:" → "is divisible". "Phrase table" is used to generate phrase pair, which contains various phrase-to-phrase relationships with probabilities. Here to maintain the grammatical structure of the natural language, reordering is performed.

Figure 2 shows the overview of PBMT technique for pseudo code generation. As shown in the figure, Phrase translation model calculates the probability Pr with respect to Target t and Source s. Reordering model calculates Probability Pr for arrangement of each phrase with respect to given Phrase pair ϕ and Alignment a. Language model finds the fluency of target sentence t.

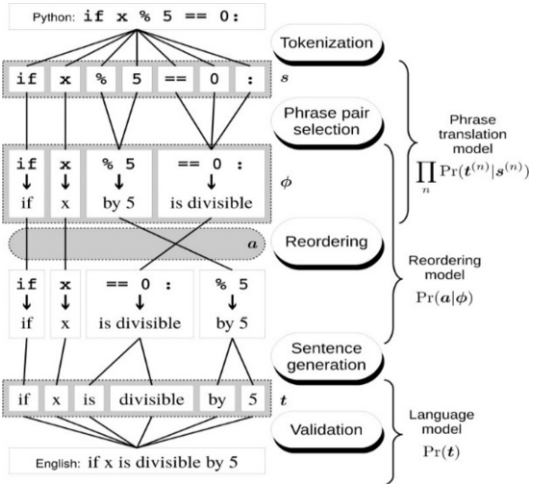

Figure 2. Python to English pseudo-code generation using PBMT technique [14]

PBMT is not able handle hierarchical correspondence of the source code and thus Tree to String Machine Translation (T2SMT) is useful. It makes use of parse tree of the input sentence. In this technique instead of phrase pair, "derivation" is introduced, where it represents relationship between source subtree and target phrase.

The above proposed approach is based on rule-based machine translation; thus, it can be generalized for variety of languages by applying corresponding rule. But this methodology does sentence wise translation due to which it is unable to handle multiple statements.

3.3. Code summarization based on Natural Language Generation

Natural language generation is a subfield of AI which translates given data into natural language such as English. In [15], source code summarization technique is utilized which generates description in English for given Java code. This approach is works by analyzing how method is invoked. In this approach, PageRank algorithm is used to find most important method in the most important context. Then Software Word Usage Model (SWUM) identifies the keywords from the action performed the important methods.

Finally, the custom Natural Language Generation (NLG) technique to generate the English descriptions which describes what methods actually do. In this NLG technique, first 6 different types of messages are created which represents different contexts of the methods. Table 1 shows different types of messages and its corresponding explanation.

Table 1. Types of messages

Message Type	Explanation
Quick Summary Message	This is short sentence which gives description of the function.
Return Message	Return type of the method is given by this type of message.
Importance Message	It shows the importance of the method based on PageRank.
Output Used Message	This is to describe maximum 2 methods which calls this method.
Call Message	This describes maximum 2 methods which is called by this method.

Next step is lexicalization, in which according to above message type phrases are used to describe it. After lexialization, more readable phrases are generated in the aggregation. Finally, senteneces are generated from the phrases generated from above step.

The above proposed approach makes use of context specific information, due to which it provides meaningful situation based or contextual output.

3.4. Tree Convolutional Neural Network (Tree CNN) based source code summarization.

Convolutional layer applied to neural network helps to extract important features from the input. In [16], Tree CNN is utilized in which program's structure is captured using

Abstract Syntax Tree (AST). AST generally captures syntactic structure of the language using hierarchical representation. Each component in the program is represented as AST's node. AST's node is denoted as a vector based on coding criterion. Then convolutional layer detects the structural features of the program. The new tree generated after convolution, has same size and shape as of original one. To solve this issue, dynamic pooling is utilized. This proposed approach basically classifies program according to corresponding functionality.

Above proposed work captures the meaning of the code snippet due to its hierarchical representation by using AST. But it exposed to training data which may cause it to suffer from the exposure bias.

3.5. Source code Markup Language (SrcML) based code summarization.

Source code Markup Language (SrcML) converts the source code of the various programming languages like Java, C, and C++ to XML file [17]. In the proposed work [18], input is the XML file for the given code snippet and output is the document file. This target document file is a combination of various parts in which every part gives description of the important part of the code snippet. SrcML considers numerous factors like white spaces, classes, parameters, and conditions and accordingly XML files is generated. The main components of the source code are represented with the help of the tags in XML like <class>, <function>, <loop> etc. Feature extractor fetches the data generated from XML file. Using XPath, queries are performed to identify each object from the code snippet that extracts four features: attributes, conditions, calls, functions. The variables or parameters included in the source code are identified as attributes. The tag<decl> is used to fetch these attributes. Conditions include "if" conditions as well as several types of loops. Number of calls performed by the source code are grouped into calls and are presented with the tag <call>. The functions with the name and the data type of the value returned by the code snippet are represented with the tag <function>. The feature extractor generates a program structure information file that can be used by code description generator. Code descriptor generator reads two files one is source code, and another is program structure information file then generates the comments based on the source code and related information.

The above proposed model makes use of tags which focuses on core components of the code, due to which complete and clear comments can be generated. Although it focuses on main concepts of code but does not show the inheritance relationships among classes. Also, proposed model only considers Object oriented aspects of the programming language and thus its useful only in case of object-oriented language.

4. Conclusion

Source code summarization technique generates the descriptions from the source code, which describes what source code intend to perform. Summary of the source code is useful for software maintenance, code search as well as code categorization.

Most of the existing source code summarization techniques were unable to capture

the structure of the source code and facing the exposure bias issue. Also, it is necessary to build the source code summarization technique that will have human evaluation criteria as well. In future, Generative Adversarial Networks (GAN) in which generator and discriminator can be combinedly designed to generate the summary which can be helpful to deal with the exposure bias issue.

References

[1] Moratanch N, Chitrakala S. A survey on extractive text summarization. International Conference on Computer, Communication and Signal Processing (ICCCSP).2017.
[2] Zhuang H, Zhang W. Generating Semantically Similar and Human-Readable Summaries With Generative Adversarial Networks.2019. IEEE Access 7:169426–169433.
[3] Alwis V . Intelligent E-news Summarization. 18th International Conference on Advances in ICT for Emerging Regions (ICTer).2018.
[4] Kini M. M, Pai K. A Survey on Video Summarization Techniques. Innovations in Power and Advanced Computing Technologies (i-PACT).2019.
[5] Ayodele T, Shoniregun CA, Akmayeva GA. Security review of email summarization systems. World Congress on Internet Security (WorldCIS-2011).2011.
[6] Kajko-Mattsson M. A Survey of Documentation Practice within Corrective Maintenance. Empirical Software Engineering. 2005; 10:31–55.
[7] Forward A, Lethbridge TC. The relevance of software documentation, tools and technologies. Proceedings of the 2002 ACM symposium on Document engineering (DocEng '02).2002.
[8] LaToza TD, Venolia G, DeLine R. Maintaining mental models. Proceeding of the 28th international conference on Software engineering (ICSE '06).2006.
[9] McMillan C, Grechanik M, Poshyvanyk D. Detecting similar software applications. 34th International Conference on Software Engineering (ICSE).2012.
[10] Anh Tuan Nguyen, & Nguyen, T. N. Automatic Categorization with Deep Neural Network for Open-Source Java Projects. IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C).2017.
[11] Sadowski, C., Stolee, K. T., & Elbaum, S.How developers search for code: A case study. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering.2015.
[12] Nie, L., Jiang, H., Ren, Z., Sun, Z., & Li, X. Query Expansion Based on Crowd Knowledge for Code Search. IEEE Transactions on Services Computing. 2016; 9(5): 771-783.
[13] Wan, Y., Zhao, Z., Yang, M., Xu, G., Ying, H., Wu, J., & Yu, P. S. Improving automatic source code summarization via deep reinforcement learning. Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering.2018.
[14] Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. Learning to generate pseudo-code from source code using statistical machine translation. 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2015.
[15] McBurney, P. W., & McMillan, C. Automatic source code summarization of context for java methods. IEEE Transactions on Software Engineering. 2016. 42(2): 103-119.
[16] Mou, L., Li, G., Zhang, L., Wang, T., & Jin, Z. Convolutional Neural Networks over Tree Structures for Programming Language Processing. Proceedings of the AAAI Conference on Artificial Intelligence. 2016; 30.
[17] Mohsin, A. H., & Hammad, M. A code summarization approach for object oriented programs. International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT).2019.
[18] Collard, M. L., Decker, M. J., & Maletic, J. I. SrcML: An infrastructure forthe exploration, analysis, and manipulation of source Code: A Tool demonstration. IEEE International Conference on Software Maintenance.2013.