Training Deep Code Comment Generation Models via Data Augmentation

Xiaoqing Zhang
zhangxq@nuaa.edu.cn
Nanjing University of Aeronautics and Astronautics
Nanjing, China

Yu Zhou
zhouyu@nuaa.edu.cn
Nanjing University of Aeronautics and Astronautics
State Key Lab. for Novel Software Technology, Nanjing University
Nanjing, China

Tingting Han
t.han@bbk.ac.uk
Birkbeck, University of London
London, UK

Taolue Chen
t.chen@bbk.ac.uk
Birkbeck, University of London
London, UK

ABSTRACT

With the development of deep neural networks (DNNs) and the publicly available source code repositories, deep code comment generation models have demonstrated reasonable performance on test datasets. However, it has been confirmed in computer vision (CV) and natural language processing (NLP) that DNNs are vulnerable to adversarial examples. In this paper, we investigate how to maintain the performance of the models against these perturbed samples. We propose a simple, but effective, method to improve the robustness by training the model via data augmentation. We conduct experiments to evaluate our approach on two mainstream sequence-sequence (seq2seq) architectures which are based on the LSTM and the Transformer with a large-scale publicly available dataset. The experimental results demonstrate that our method can efficiently improve the capability of different models to defend the perturbed samples.

CCS CONCEPTS

• **Software and its engineering**; • **Computing methodologies** → **Artificial intelligence**;

KEYWORDS

deep neural networks, code comment generation, data augmentation

ACM Reference Format:
Xiaoqing Zhang, Yu Zhou, Tingting Han, and Taolue Chen. 2020. Training Deep Code Comment Generation Models via Data Augmentation. In *12th Asia-Pacific Symposium on Internetware (Internetware'20), May 12–14, 2021, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3457913.3457937

1 INTRODUCTION

In software and system development, automated processing and analysis of program source code can greatly reduce the developing cost, which has received tremendous interests from industry. It is not until several years ago, researchers started to propose using deep neural networks (DNNs) for the task of source code processing. In particular, encouraging process has been made in code comment generation tasks [1, 8, 19, 21, 24].

Szegedy et al. [18] first observe that adding imperceptible perturbations on the original inputs can fool the state-of-the-art DNNs used for image classification, which are referred to as adversarial examples. The existence of adversarial examples may seriously affect the reliability and security of the deep learning models which discourages their application in practice. In recent years, many efforts in computer vision (CV) and natural language processing (NLP) have concentrated on how to improve the robustness the DNN models against the prevalent adversarial examples.

Very recently, some researchers [3, 17, 22, 23] have investigated the robustness of neural networks applied to programming, including code classification models, method name prediction models and type prediction models. However, the robustness of code comment generation models has received little attention. In this paper, we focus on the code comment generation task which is more akin to the neural machine translation in the NLP domain and explore the robustness of DNNs used for this task. We propose to adopt a well-studied technique for robustness enhancement, i.e., adversarial training based on data augmentation to improve the robustness. We carry out experiments on the publicly available source code dataset to assess the robustness of two code comment generation models, a Transformer-based sequence-to-sequence (seq2seq) model and an LSTM-based seq2seq model. The experiment results show that our approach can defend the model against adversarial attacks without modifying the model structure and the training procedure. Our code and data are publicly available[1].

[1]https://github.com/Zhangxq-1/coderepository.git

The remainder of this paper is structured as follows. Section 2 presents the related work. Section 3 describes our proposed approach. Section 4 introduces the evaluation setup and results. Section 5 discusses the threats to validity. Section 6 concludes our work.

2 RELATED WORK

Since Szegedy et al. [18] first observed that deep neural networks are vulnerable to small perturbation, a lot of efforts have emerged to study the robustness of DNNs. In CV, they proposed the first adversarial examples method named L-BFGS to test the robustness of the image classification model. Goodfellow et al. [6] presented a simplified method FGSM which is a gradient-based attack method. Besides, there are some other representational works [4, 12]. Researchers try to employ approaches in the image domain to texts and propose some efficient approaches. For example, in text, Papernot et al. [15] proposed the first adversarial examples generation algorithm using the unfolding computational graph. Instead of gradient loss, Javid et al. [9] generated adversarial examples to attack the classification model by performing the atomic flip.

In neural program analyzers, Wang et al. [20] first compared the robustness of different program representation towards the semantics prediction task. Zhang et al. [23] exploited the MHM approach to generate adversarial examples towards the code classification models. Pavol et al. [3] proposed a novel technique to address the problem of learning accurate and robust models of the type prediction task. Ramakrishnan et al. [17] and Yefet et al. [22] focused on the robustness of the method name prediction model.

To improve the robustness of DNN models, existing methods can be classified into two categories: adversarial training which contains data augmentation [10], model regularization [14], and robust optimization [12], as well as knowledge distillation [16].

3 APPROACH

In this section, we present our method for improving robustness. The overview of our approach is given in Figure 1. It consists of four steps: (1) training and testing the model with the original dataset; (2) generating the perturbed test dataset to test the robustness of the trained model; (3) generating the data augmentation training dataset to retrain the model; (4) testing the retrained model on both the original set and perturbed set.

Our goal is to learn robust deep code comment generation models that can maintain the performance of models under perturbed inputs. Thus, the first question is how to generate the perturbations for the original test dataset. To this end, we employ the MHM [23] algorithm which is the state-of-the-art method of perturbing source code by renaming identifiers. MHM takes all the programmer-defined identifiers in the source code as tokens to be substituted and renames them iteratively according to Metropolis-Hastings (M-H) sampling [5, 7, 13]. At each iteration, a change from x to x' is accepted based on the acceptance rate:

$$\alpha(x'|x) = min\{1, \alpha^*\}$$

where

$$\alpha^* \approx \frac{1 - C(x')[y]}{1 - C(x)[y]}$$

In this paper, $C(x)$ is instantiated as the BLEU score.

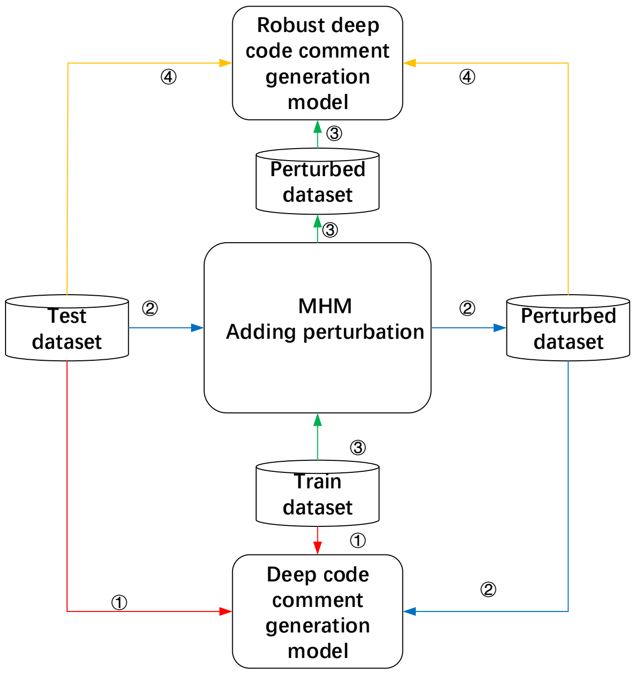

Figure 1: The workflow of our approach

In order to enhance the robustness of deep code comment generation models, we adopt the adversarial training method based on data augmentation, a simple but effective method. For the examples $\langle p, comment \rangle$ in the training set D^{tra}, we first randomly sample a subset $D^{tra}_{sub} \subseteq D^{tra}$ which contains Num examples ($Num = 9,957$ in our experiment). For each sample in D^{tra}_{sub}, we generate adversarial example $\langle p_{adv}, comment \rangle$ according to the MHM algorithm, and then mix the adversarial examples with the original dataset to generate a new dataset D^{tra}_{adv}, and retrain the model using D^{tra}_{adv} without changing the original model's parameters. In this way, we can obtain a more robust model.

4 EVALUATION

4.1 Dataset

We conduct experiments based on the publicly available Java source code dataset [8] which contains methods extracted from Java projects on GitHub.[2] This dataset provides method-level source code and comment. We only extract the first sentence of Javadoc as the comment, which represents the functionality of the method. We preprocess the dataset by the Javalang parser[3] to extract identifiers. The statistics of the dataset are shown in Table 1.

4.2 Target Models

We adopt two commonly used seq2seq models based on the LSTM and Transformer [1] architectures as the target models in our experiments to study how to improve the robustness of deep code comment generation models. We reuse the code from the original

[2]https://github.com
[3]https://github.com/c2nes/javalang

Table 1: Information about the Java dataset.

Dataset	Java
Train	**69708**
Validation	**8714**
Test	**8714**
Avg identifiers in Dataset	**3.2**

work and follow their instructions to set hyper-parameters. Both models are seq2seq models at the text level. The input of these models is token sequences, which are obtained by splitting the original source code. The LSTM architecture with attention is widely used in the field of source code processing, and to the best of our knowledge, the Transformer architecture represents the state-of-the-art result on the Java dataset.

4.3 Metrics

Robustness can be evaluated by the performance changes of DNNs before and after the adversarial attack. BLEU, METEOR, and ROUGE-L are three automatic machine translation metrics which have been widely adopted to measure the quality of the generated comments in code comment generation tasks.

- BLEU. The BLEU score refers to the degree of similarity between the generated text and the reference texts, with values closer to 1 representing more similarity.
- METEOR. METEOR evaluates the translation result based on recall and accuracy measures, aligns it with the reference, and calculates the sentence level similarity score.
- ROUGE-L. ROUGE-L focuses on recall score which computes similarity based on the longest common subsequence between the translation result and the reference.

A robust model should be stable and should not change its output much when exposed to adversarial examples. Compared with the performance on the original test dataset, the lower the value after the adversarial attack implies the worse robustness of the model. Therefore, we use these three metrics to evaluate the robustness of the model.

4.4 Results

In our experiment, we use the perturbed examples replacing K identifiers (in our experiment, we set $K = 3$) to study the robustness of different deep code comment generation models by observing the changes of the BLEU, METEOR, and ROUGE-L values. The experimental results on LSTM and Transformer architectures are shown in Table 2.

It can be observed from Table 2 that the performance of both models significantly degrades with the perturbed dataset. Between these two models, even though the results of the Transformer model on the clean dataset are better, relative degradation of the BLEU score, METEOR score, and ROUGE-L score (66.42%, 65.30%, and 53.27%) on the perturbed dataset are worse than the LSTM model with relative degradation of 44.60%, 48.28% and 39.06%. (Note that a larger value of relative degradation means worse robustness.) It can be seen that the performance of the model cannot be guaranteed to maintain when it is subject to perturbation attacks.

Table 2: Results on the perturbed test dataset with replacing K identifiers. ('Clean' means the original test dataset and 'Adv' is the perturbed test dataset.)

		Clean	Adv
Transformer	**BLEU**	**44.58**	**14.97**
	METEOR	**26.43**	**9.17**
	ROUGR-L	**54.76**	**25.59**
LSTM	**BLEU**	**35.47**	**19.65**
	METEOR	**19.72**	**10.2**
	ROUGE-L	**47.57**	**28.99**

To validate the effectiveness of the data augmentation based adversarial training method, new models without changing the hyper-parameters and model structures are trained from scratch on the adversarially augmented training set which is mixed with the perturbed examples. Then, we test the new model with the original test dataset (i.e., 'Clean' in Table 3) and the perturbed test dataset with K identifiers ($K = 3$) replaced (i.e., 'Adv' in Table 3). In Table 3, 'Nor' represents the model through the standard training process, and 'Aug' represents the model trained by data augmentation.

It has been confirmed that improving robustness may sacrifice the accuracy of the models on the clean dataset [2, 11]. We can observe from Table 3 that the BLEU, METEOR and ROUGE-L scores are all significantly increased on both the original 'Clean' test dataset and the 'Adv' test dataset. It suggests that, after data augmentation, we obtain a new model which retains the accuracy but is more robust.

Table 4 shows an example of the generated summary of the Transformer model. The summary is irrelevant to the reference ('Ref' in Table 4) under adversarial attacks, whereas after the data augmentation, the generated result is closer to the reference.

5 THREATS TO VALIDITY

Internal Validity. The randomness of selecting the substituted identifier and the implementation errors may affect the reproducibility of the experiment results. To alleviate this threat, we conduct a manual inspection using an example and re-implement the experiment for three times.

External validity. We only concentrate on improving the roubstness of the comments generation model for Java methods in our approach. However, our approach is essentially programming language independent and can be applied to any comments generation model.

6 CONCLUSION

In this paper, we adopt the MHM algorithm to generate the perturbed examples to evaluate the performance of different deep code comment generation models built on LSTM and Transformer architectures. We propose a robust enhancement method for these models and conduct experiments to validate the effectiveness of the data augmentation adversarial training method. Experiment results show that our method can enhance the models' defense against attacks. In the future, we will explore more effective methods to further improve the model robustness.

Table 3: Results of the data augment adversarial training. (The more BLEU increases on the 'Adv' dataset, the more robust the model is after adversarial training.)

		BLEU	METEOR	ROGUE-L
Transformer	Nor-Clean	44.58	26.34	54.76
	Nor-Adv	14.97	9.17	25.59
	Aug-Clean	45.41	27.31	54.78
	Aug-Adv	26.77	15.75	36.97
LSTM	Nor-Clean	35.47	19.72	47.57
	Nor-Adv	19.65	10.2	28.99
	Aug-Clean	39.07	21.69	49.98
	Aug-Adv	25.40	13.62	35.27

Table 4: An example of adversarial attack before and after data augmentation.

Clean	static boolean isPackageAccess (final int modifiers) { return (modifiers & ACCESS_TEST) == _NUM; }
Adv	static boolean statistic (final int modifiers) { return (modifiers & ACCESS_TEST) == _NUM; }
Ref	returns whether a given set of modifiers implies package access.
Nor-Adv	returns true if thistype, returns false if the given modifiers.
Aug-Adv	learn whether a given set of modifiers implies package access.

ACKNOWLEDGMENTS

This work was partially supported by the National Key R&D Program of China (No. 2018YFB1003902), the National Natural Science Foundation of China (NSFC, No. 61972197), the Natural Science Foundation of Jiangsu Province (No. BK20201292), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Qing Lan Project. T. Chen is partially supported by NSFC grant (No. 61872340), and Guangdong Science and Technology Department grant (No. 2018B010107004), Natural Science Foundation of Guangdong Province, China (No. 2019A1515011689).

REFERENCES

[1] Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A Transformer-based Approach for Source Code Summarization. *arXiv preprint arXiv:2005.00653* (2020).

[2] Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173* (2017).

[3] Pavol Bielik and Martin T. Vechev. 2020. Adversarial Robustness for Code. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event.* 896–907.

[4] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017.* 39–57.

[5] Chib, Siddhartha, Greenberg, and Edward. 1995. Understanding the Metropolis-Hastings Algorithm. *American Statistician* (1995).

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICML.*

[7] W. Keith Hastings. 1970. Monte Carlo Sampling Methods Using Markov Chains and Their Application. *Biometrika* 57, 1 (1970).

[8] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. 2018. Summarizing Source Code with Transferred API Knowledge. In *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18.*

[9] Ebrahimi Javid, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751* (2017).

[10] Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017.* 2021–2031.

[11] Pengcheng Li, Jinfeng Yi, Bowen Zhou, and Lijun Zhang. 2019. Improving the Robustness of Deep Neural Networks via Adversarial Training with Triplet Loss. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019.* 2909–2915.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings.*

[13] N. METROPOLIS. 2004. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* 21 (2004).

[14] Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial Training Methods for Semi-Supervised Text Classification. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings.*

[15] Nicolas Papernot, Patrick D. McDaniel, Ananthram Swami, and Richard E. Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. In *2016 IEEE Military Communications Conference, MILCOM 2016, Baltimore, MD, USA, November 1-3, 2016.* 49–54.

[16] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016.* 582–597.

[17] Goutham Ramakrishnan, Jordan Henkel, Zi Wang, Aws Albarghouthi, Somesh Jha, and Thomas W. Reps. 2020. Semantic Robustness of Models of Source Code. abs/2002.03043 (2020). https://arxiv.org/abs/2002.03043

[18] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv: Computer Vision and Pattern Recognition* (2013).

[19] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S. Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018.* 397–407.

[20] Ke Wang and Mihai Christodorescu. 2019. COSET: A Benchmark for Evaluating Neural Program Embeddings. *CoRR* abs/1905.11445 (2019).

[21] Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code Generation as a Dual Task of Code Summarization. In *Advances in Neural Information Processing Systems 32.* 6563–6573.

[22] Noam Yefet, Uri Alon, and Eran Yahav. 2020. Adversarial examples for models of code. *Proc. ACM Program. Lang.* 4, OOPSLA (2020).

[23] Huangzhao Zhang, Zhuo Li, Ge Li, Lei Ma, Yang Liu, and Zhi Jin. 2020. Generating Adversarial Examples for Holding Robustness of Source Code Processing Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1169–1176.

[24] Yu Zhou, Xin Yan, Wenhua Yang, Taolue Chen, and Zhiqiu Huang. 2019. Augmenting Java method comments generation with context information based on neural networks. *Journal of Systems and Software* 156 (2019), 328–340.