TARGET: Pre-trained Transformer for Contextualized Code Commit Message Generation

Anonymous ACL submission

Abstract

Commit messages are essential as the natural language documentation of source code changes throughout software development cycle, and automatically generating high-quality commit messages can substantially release developers' burden of writing ones over the frequent updates. However, the semantic gap between source code and natural language lies as the major challenge in front of the task. Several studies have been proposed to alleviate the challenge but none took code contextual information into consideration. In this paper, we put forward a novel generation model with pre-training to obtain the contextual representation of source code. Experiments on the benchmark demonstrate the superior effectiveness of our model than the state-of-art models, with at least 16.13% increase in BLEU-4 score. Furthermore, we also highlight the future opportunities in more accurate code commit message generation, such as enriching the semantics of code changes using API documentation.

1 Introduction

Transferring the semantics from source code to natural language and vice-versa is a practically useful but challenging task. The essential challenge lies in the semantic gap between source code and natural language texts. Mitigating the gap can be extremely beneficial for many tasks, such as code commit message generation. In the life cycle of software development, commit messages are essential for developers to document the *code changes* (or called *code difference*) in natural language when committing an update to the version control systems. High-quality messages can ease the developers' comprehension of the software evolution without diving into the implementation details.

However, the quality of code commit messages is not guaranteed in practice. Developers generally neglect the writing of commit messages due to the lack of time and incentive. Therefore, automatic generation of commit messages becomes necessitated and many approaches have been proposed to address the need. At the earlier stage, researchers adopt pre-defined templates to generate commit messages (Buse and Weimer, 2010; Cortes-Coy et al., 2014; Vásquez et al., 2015; Shen et al., 2016). Later works leverage information retrieval techniques to reuse the existing commit messages based on similarity (Huang et al., 2017; Liu et al., 2018). With the advancement of neural machine translation (NMT), recent researchers treat the commit message generation as a text translation task and utilize the deep learning models for commit message generation, which are claimed to achieve the state-of-the-art performance on the benchmark (Jiang et al., 2017; Loyola et al., 2017; Xu et al., 2019; Liu et al., 2019). Despite the success of the NMT models in commit message generation, these studies all adopt static embedding to encode code changes and neglect the contextual information in code structure.

In this paper, we propose a novel approach, named TARGET, that exploits contextual information of code changes from a pre-trained Transformer mechanism. Based on the pre-trained results, TARGET fine-tunes the generation model by retaining the contextual representations in the encoder and further optimizing the whole model through supervised training.

Experimental results show that TARGET outperforms the state-of-the-art approaches on all evaluation metrics on the benchmark dataset. The effectiveness of combining contextual information has also been demonstrated.

In summary, the contributions of our work are as follows: 1) We propose a novel code commit message generation model with contextual information of code changes enriched; 2) We conduct extensive experiments to evaluate the performance of the pro-

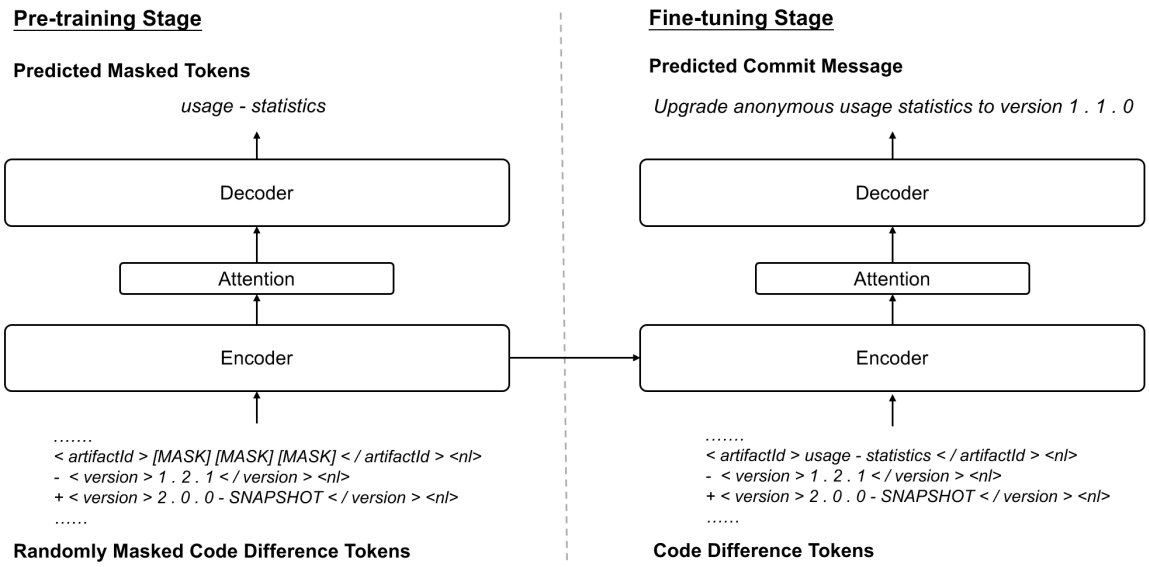

Figure 1: Architecture of TARGET. We adopt a two-stage approach by first pre-training the encoders with a bi-directional language model to learn contextual information, then fine-tuning the whole model with labelled dataset

posed model; and 3) We highlight the opportunities for more accurate code commit message generation and will release our implementation details for usage by future researchers.

2 Related Work

In this section, we review the studies most relevant to our work. We group them into two categories: 1) Code commit message generation; 2) Contextual word embedding in natural language processing.

2.1 Code Commit Message Generation

According to the existing literature, we can roughly separate the methods of commit message generation into three categories: Pre-defined template based, similarity based and deep-learning based.

In the first category, the researchers (Buse and Weimer, 2010; Cortes-Coy et al., 2014; Vásquez et al., 2015; Shen et al., 2016) translate the code difference into natural language according to the pre-defined templates. But these methods can only handle the code differences which match the pre-define templates.

Similarity-based methods (Huang et al., 2017; Liu et al., 2018) are to generate commit messages through searching similar code changes and reusing their commit messages. But these methods heavily depend on whether the most relevant changes can be searched in the corpus.

Deep-learning-based methods (Jiang et al., 2017; Xu et al., 2019; Loyola et al., 2017, 2018) treat code commit message generation task as a translation task between two language types, *i.e.*, code changes and commit messages. But in these methods, the embedding of code tokens do not explicitly take advantage of the contextual information in source code.

2.2 Contextual Word Embedding in Natural Language Processing

In recent year, a significant portion of work in natural language processing studies how to obtain a good representation of a word or sentence. Contextual word embedding is an unsupervised pre-training method which aims at finding a good starting point for training. Different from the static embedding methods (Turian et al., 2010; Mnih and Hinton, 2008; Mikolov et al., 2013), contextual embedding methods (Devlin et al., 2018; Radford et al., 2018, 2019; Song et al., 2019) also take the contextual meanings of words into account, during which the representation of tokens is pre-trained based on unlabeled texts and then fine-tuned for the downstream task. Compared to the static embedding methods that fail to capture polysemy, contextual embedding methods capture the word semantics in different contexts and represent the word tokens more accurately.

3 Methodology

We propose TARGET, a pre-trained Transformer that transfers the internal representations from bi-

directional language model for contextual code embedding to generate the code commit messages from source code differences. Figure 1 gives an overview of our model. The input X, which is a sequence of source code differences, is first represented as a sequence of embedding vectors. In the pre-training stage, the Transformer encoder captures the contextual information for each token via training a bi-directional language model, similar to the previous works of BERT and MASS (Devlin et al., 2018; Song et al., 2019). Then in the fine-tuning stage, the internal states of encoder layers are transferred for downstream code commit message generation task.

3.1 Pre-training

We pre-train a bi-directional Transformer to map the input source code difference tokens into a sequence a contextual embedding vectors. Given a source code difference sequence x from the dataset X, we randomly mask its fragment from position u to v, denoted as $x^{u:v}$. Then TARGET pre-trains the Transformer by predicting the masked fragment $x^{u:v}$ from the masked input sequence $x^{\backslash u:v}$. Log likelihood is used as the objective function:

$$L(\theta; X) = -\frac{1}{|X|}\Sigma_{x \in X} \log P(x^{u:v}|x^{\backslash u:v}; \theta) \quad (1)$$

By training a bi-directional language model that requires the Transformer to predict the masked code difference tokens from their context, contextual information is incorporated into the encoders, which produces the contextual code representations. Contextual code embedding can capture high-level information from the rest tokens in the code sequence, such as long-term dependencies and code structure. After the unsupervised pre-training on the code difference corpus, the contextual representations in the encoders are transferred to the code commit message generation task for supervised fine-tuning.

3.2 Fine-Tuning

In the fine-tuning stage, TARGET reserves the hidden states of the encoders for contextual code embedding. Both the encoder and decoder are optimized throughout the supervised fine-tuning with back-propagation applied to all layers. Given a source code difference sequence x, the entire model is fine-tuned to predict the corresponding code commit message sequence m.

Code Difference	mmm a / pom.xml ppp b / pom.xml < dependency> <groupId> org.graylog.plugins </groupId> <artifactId> usage - statistics </artifactId> - <version> 1.2.1 </version> + <version> 2.0.0 - SNAPSHOT </version> <scope> provided </scope> </dependency>
NMT	Bump version to 2 . 1 . 0 - SNAPSHOT
NNGen	Update usage plugin
PtrGNCMsg	Updated version to 2 . 0 . 0 - SNAPSHOT
TARGET	Upgrade anonymous usage statistics to version 2 . 0 . 0
Reference	Upgrade to anonymous usage statistics plugin 2 . 0 . 0 - SNAPSHOT

Figure 2: Example for illustrating the effectiveness of TARGET.

Example1	
Code Difference	new file mode 100644 index 0000000 . . da7d70d mmm / dev / null ppp b / WindowsPhone / README . md + * * PLEASE NOTE * * : Unless * * explicitly * * stated , most of these plugins will * not * work with Cordova / PhoneGap 3 . x . x out of the box . They will need updating before they can be used via the ` plugin add ` interface . +
TARGET	Added a readme . md
Reference	Added a note about 3 . 0 . 0 compatibility
Example2	

Code Difference	mmm a / RCTVideo . m
	ppp b / RCTVideo . m
	static NSString * const playbackRate = @ " rate " ;
	-(void) removeFromSuperview
	{
	[_player pause] ;
	+ if (_playbackRateObserverRegistered) {
	[_player removeObserver : self forKeyPath : playbackRate] ; _playbackRateObserverRegistered = NO ;
	+ }
	_player = nil ;
	[self removePlayerLayer] ;

TARGET	Restore missing property name
Reference	Remove observer only if it has been registered

Figure 3: Examples for illustrating the error cases generated by TARGET.

4 Experimental Analysis

4.1 Dataset

The dataset we use is from Jiang et al. (2017)'s work which contains 2 million commits collected from popular Java projects in GitHub. Following (Liu et al., 2018), the dataset preparation process includes two stages, *i.e.*, preprocessing and filtering. In preprocessing step, we remove some useless part in dataset, such as commit id and issue id. And in filtering step, we remove those messages not satisfied with some pre-defined conditions. After data cleaning, the dataset remained 32K commit message for our task.

4.2 Baselines

We compare several baseline models. The first baseline is NMT (Jiang et al., 2017), which considers the source code differences and code commit messages as two different languages, thus uses an atentional RNN Encoder-Decoder framework to

Table 1: Comparison of our model with baselines using different evaluation metrics. TARGET $_{\text{No pre-training}}$ refers to the ablation model without the bi-directional language model pre-training stage. TARGET $_{\text{No fine-tuning}}$ refers to the ablation model with encoder's parameters frozen and not optimized in the fine-tuning stage

Model		BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Baselines	NMT	14.17	21.29	12.19	20.85	12.99
	NNGen	16.43	25.86	15.52	24.46	14.03
	PtrGNCMsg	9.78	23.66	9.61	23.67	11.41
Ours	TARGET $_{\text{No pre-training}}$	18.82	29.36	17.58	27.70	15.05
	TARGET $_{\text{No fine-tuning}}$	18.09	29.17	17.54	27.55	14.65
	TARGET	**19.08**	**30.45**	**17.94**	**28.69**	**15.34**

translate the code differences to commit messages. Comparatively, the second baseline NNGen (Liu et al., 2018) leverages the nearest neighbour algorithm to retrieve the most similar commit messages. It represents the code differences as vectors by a bags of words model and computes the cosine similarity between the queried code differences with the existing one to measure the similarity. The third baseline is PtrGNCMsg (Liu et al., 2019), which uses a pointer-generator network proposed by See et al. (2017), to copy the out-of-vocabulary context-specific words from the source sentence.

4.3 Results

The comparison results are shown in Table 1. TARGET outperforms the baselines in terms of all metrics, providing evidence for the effectiveness of source code contextual embedding by pre-training.

We also compare TARGET with two ablation methods, a Transformer learnt from scratch without pre-training stage for deriving contextual information, and a pre-trained Transformer without fine-tuning. It can be seen that both methods perform worse than TARGET, demonstrating the effectiveness of the two-stage training in TARGET. It is also noteworthy that the pre-trained TARGET with encoder's parameters frozen in fine-tuning stage still outperforms the baselines significantly, which can be attributed to the contextual information captured in the pre-training stage.

4.4 Case Study

One example of the generated commit messages from different models is displayed in Figure 2. As can be seen, our model generates an obviously better result which is more consistent with the reference. The other three baselines fail to infer the important words, such as "*Anonymous*","*statistics*". However, our method can well predict the words based on the contextual semantics of the code change. Due to the page limit, we will list more examples later to improve the paper draft.

4.5 Error Analysis

The errors are mainly for two cases based on our observation.

Lack of details in commit messages As shown in example 1 in Figure 3, a human-written commit message not only describes the modifications but also explains the reasons. However, the commit messages generated by TARGET plainly describes the change action without explanation, which can increase developers' burden in comprehending the code changes. This can be solved in the future by training with commit messages with better quality and penalizing the short commit messages in generation with a better-designed objective function.

Missing information of API/function names As shown in example 2 in Figure 3, TARGET fails to capture the information inside the API names or function names when generating the code commit messages, which are critical in understanding the code changes. To fix this, the abstract syntax tree can be incorporated to exploit the information inside API functions.

5 Conclusions and Future Work

In this paper, we propose TARGET, a novel model that first incorporates the contextual information of code differences by pre-training a bi-directional language model. Comparison with the baselines has shown that TARGET achieves the new state-of-the-art. In future, we plan to incorporate API documentation to well capture the semantics of API/function names for more accurate code commit message generation.

References

Raymond P. L. Buse and Westley Weimer. 2010. Automatically documenting program changes. In *ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010*, pages 33–42.

Luis Fernando Cortes-Coy, Mario Linares Vásquez, Jairo Aponte, and Denys Poshyvanyk. 2014. On automatically generating commit messages via summarization of source code changes. In *14th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2014, Victoria, BC, Canada, September 28-29, 2014*, pages 275–284.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yuan Huang, Qiaoyang Zheng, Xiangping Chen, Yingfei Xiong, Zhiyong Liu, and Xiaonan Luo. 2017. Mining version control system for automatically generating commit comment. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017*, pages 414–423.

Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, pages 135–146.

Qin Liu, Zihe Liu, Hongming Zhu, Hongfei Fan, Bowen Du, and Yu Qian. 2019. Generating commit messages from diffs using pointer-generator network. In *Proceedings of the 16th International Conference on Mining Software Repositories*, pages 299–309. IEEE Press.

Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: how far are we? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, pages 373–384.

Pablo Loyola, Edison Marrese-Taylor, Jorge A. Balazs, Yutaka Matsuo, and Fumiko Satoh. 2018. Content aware source code change description generation. In *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*, pages 119–128.

Pablo Loyola, Edison Marrese-Taylor, and Yutaka Matsuo. 2017. A neural architecture for generating natural language descriptions from source code changes. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 287–292.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Jinfeng Shen, Xiaobing Sun, Bin Li, Hui Yang, and Jiajun Hu. 2016. On automatic summarization of what and why information in source code changes. In *40th IEEE Annual Computer Software and Applications Conference, COMPSAC 2016, Atlanta, GA, USA, June 10-14, 2016*, pages 103–112.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394.

Mario Linares Vásquez, Luis Fernando Cortes-Coy, Jairo Aponte, and Denys Poshyvanyk. 2015. Changescribe: A tool for automatically generating commit messages. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, pages 709–712.

Shengbin Xu, Yuan Yao, Feng Xu, Tianxiao Gu, Hanghang Tong, and Jian Lu. 2019. Commit message generation for source code changes. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3975–3981.