2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering (Forge)

Commit Message Generation via ChatGPT: How Far Are We?

Yifan Wu
Peking University
Beijing, China
yifanwu@pku.edu.cn

Ying Li*
Peking University
Beijing, China
li.ying@pku.edu.cn

Siyu Yu
The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen)
Shenzhen, China
gaiusyu6@gmail.com

ABSTRACT

Commit messages concisely describe code changes in natural language and are important for software maintenance. Various automatic commit message generation approaches have been proposed, such as retrieval-based, learning-based, and hybrid approaches. Recently, large language models have shown impressive performance in many natural language processing tasks. Among them, ChatGPT is the most popular one and has attracted wide attention from the software engineering community. ChatGPT demonstrates the ability of in-context learning (ICL), which allows ChatGPT to perform downstream tasks by learning from just a few demonstrations without explicit model tuning. However, it remains unclear how well ChatGPT performs in the commit message generation task via ICL. Therefore, in this paper, we conduct a preliminary evaluation of ChatGPT with ICL on commit message generation. Specifically, we first explore the impact of two key settings on the performance of ICL on commit message generation. Then, based on the best settings, we compare ChatGPT with several state-of-the-art approaches. The results show that a carefully-designed demonstration can lead to substantial improvements for ChatGPT on commit message generation. Furthermore, ChatGPT outperforms all the retrieval-based and learning-based approaches in terms of BLEU, METEOR, ROUGE-L, and Cider, and is comparable to hybrid approaches. Based on our findings, we outline several open challenges and opportunities for ChatGPT-based commit message generation.

CCS CONCEPTS

• **Software and its engineering** → **Software maintenance tools**.

KEYWORDS

Commit Message Generation, Large Language Model, In-Context Learning

ACM Reference Format:
Yifan Wu, Ying Li, and Siyu Yu. 2024. Commit Message Generation via ChatGPT: How Far Are We?. In *AI Foundation Models and Software Engineering (FORGE '24), April 14, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3650105.3652300

*Corresponding author.

ACM ISBN 979-8-4007-0609-7/24/04
https://doi.org/10.1145/3650105.3652300

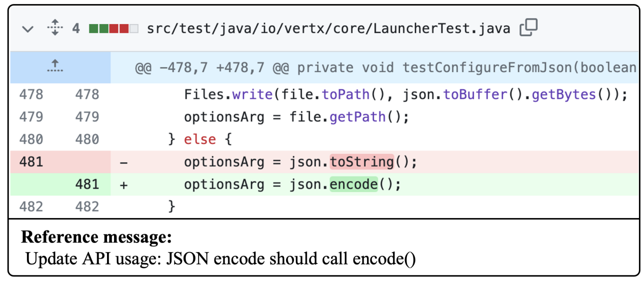

Figure 1: An example of a commit with a code change and its reference message.

1 INTRODUCTION

When submitting a code change to a version control system, developers can write a brief descriptive comment in a textual format, called a commit message. As shown in Figure 1, a commit mentioned in this paper refers to the pair of a code change and its commit message. Given a commit, we refer to its original commit message as the reference message. High-quality commit messages can greatly facilitate software maintenance by providing a human-readable summary of the changes, obviating the need for a detailed examination of the complex code and potentially simplifying code review and related tasks [27, 38]. Conversely, poor commit messages can negatively affect software defect proneness [19].

However, the inherent complexity of commits makes manually summarizing them into concise messages difficult. Furthermore, given the rapid pace of today's software development, manually writing high-quality commit messages becomes a time-intensive and arduous task [8, 26, 39]. Consequently, many approaches have been proposed to automatically generate high-quality commit messages from code changes. Early studies [4, 5, 21, 34] extract information from code changes and generate commit messages with predefined rules or templates, which may not encompass all scenarios or deduce the intent behind code changes. Later, some studies [15, 16, 24] adopt information retrieval (IR) approaches to reuse commit messages of similar code changes. They can take advantage of similar examples, but the reused commit messages may not correctly describe the content or intent of the current code change. Recently, a large number of deep learning (DL)-based commit message generation approaches have been proposed [6, 14, 23, 29, 35, 37, 41]. They trained on a large-scale commit corpus to translate code changes into commit messages and have demonstrated to outperform rule-based approaches and IR-based approaches. However, these approaches require either training models from scratch [6, 23, 29, 41] or tuning a pre-trained language model (e.g., CodeT5

[42]) with labeled data [14, 35, 37], which could be impractical due to the scarcity of computing resources and labeled data.

More recently, large language models (LLMs), which are large-sized pre-trained language models with tens or hundreds of billions of parameters and train on extensive unlabeled corpora via self-supervised learning, exhibit strong capacities to understand natural language and solve various tasks [48]. In addition to natural language, LLMs can also deal with code, which arouses growing interest in applying LLMs to the software engineering domain [50]. Among LLMs, ChatGPT [31] has attracted great attention. Compared with previous DL approaches, ChatGPT demonstrates a powerful ability of in-context learning (ICL) [3, 7], which allows ChatGPT to perform downstream tasks by learning from just a few demonstrations without explicit model tuning. Some recent works [9, 25, 37, 47] have investigated LLM-based commit message generation. However, it remains unclear how well ChatGPT performs in the commit message generation task via ICL. More research is needed to determine its ability in this important area.

Therefore, in this paper, we conduct a preliminary evaluation of ChatGPT with ICL on commit message generation using a popular multilingual dataset called MCMD [36]. We first explore the impact of two key settings on the performance of ICL in the commit message generation task: the number and selection of demonstrations. Then, based on the best settings, we compare ChatGPT with several state-of-the-art (SOTA) approaches. Our experimental results show that a carefully-designed demonstration can lead to substantial improvements for ChatGPT on commit message generation. Specifically, when the number of demonstrations is 32 and the commits which are most similar to the target one are selected as demonstrations, the BLEU, METEOR, ROUGE-L, and Cider values of ChatGPT can be increased by 124.3%, 69.5%, 89.7%, and 275.0%, respectively, compared with vanilla in-context learning. Furthermore, ChatGPT outperforms all the IR-based and DL-based approaches in terms of BLEU, METEOR, ROUGE-L, and Cider, and is comparable to hybrid approaches. Based on the findings, we outline several challenges and opportunities for ChatGPT-based commit message generation that remain to be addressed.

In summary, this paper makes the following contributions:

- To the best of our knowledge, we are the first to evaluate ChatGPT with ICL on commit message generation using a multilingual dataset and compare it with SOTA approaches.
- Based on our findings, we outline several challenges and opportunities for ChatGPT-based commit message generation.
- The code in this study is publicly available at https://github.com/wuyifan18/ChatGPT4CMG to benefit both practitioners and researchers in the field of commit message generation.

2 STUDY DESIGN

2.1 Research Questions

This study aims to investigate the effectiveness of ChatGPT on commit message generation using in-context learning. To this end, we propose to answer the following research questions.

- **RQ1: What is the effectiveness of ChatGPT on commit message generation using zero-shot, one-shot, and few-shot learning?** In the first RQ, we investigate how effective vanilla ICL (i.e., random-based demonstration selection) is on the MCMD dataset [36]. The results can also reflect to what extent the number of demonstrations (i.e., zero-shot, one-shot, and few-shot) affects the effectiveness.
- **RQ2: Can the effectiveness of ChatGPT be improved by retrieval-based demonstration selection?** Recent studies have demonstrated that the quality of demonstrations can significantly impact the effectiveness of ICL [11, 12, 22, 28]. Inspired by these studies, we investigate whether retrieval-based demonstration selection can improve ChatGPT's performance on commit message generation.
- **RQ3: How does the effectiveness of ChatGPT compare to the state-of-the-art approaches on commit message generation?** In this RQ, we aim to compare ChatGPT with eight state-of-the-art commit message generation approaches that have been trained on the same dataset (i.e., the MCMD dataset).

2.2 Prompt Design

A formatted natural language prompt is used as the input for ChatGPT to generate the commit message. Formally, a prompt is defined as $\mathcal{P} = \{\mathcal{NL} + \mathcal{CD} + x_q\}$, where $\mathcal{NL}$ is a natural language instruction to describe the commit message generation task, $\mathcal{CD} = \{(x_i, y_i)\}_{i=1}^{n}$ is a set of demonstrations composed by input code change x_i and desired commit message y_i, and x_q is a code change query to be answered by ChatGPT. $\mathcal{NL}$ used in our study is "Generate a concise commit message that summarizes the content of code changes." Such a prompt can let ChatGPT gain task-specific knowledge by learning the pattern hidden in the demonstration of the commit message generation task. Specifically, if $n = 0$ which means there is no demonstration, the setting is known as *zero-shot learning*; if $n = 1$ which means there is only one demonstration, the setting is known as *one-shot learning*; and *few-shot learning* means there are several demonstrations. Also, there is a constraint that $size(\mathcal{P}) \leq$context-window, which means the prompt should fit within the context window limit of ChatGPT.

2.3 Demonstration Retrieval

Note that the demonstrations used in RQ1 are randomly selected from the training set. In RQ2, we aim to investigate whether retrieval-based demonstration selection can enhance the effectiveness. The most widely-used method to retrieve similar code is focusing on the overlap to the code tokens [13, 18, 46]. Inspired by these studies, we utilize the Jaccard Coefficient [30] to calculate the similarity at the token level as follows: $sim(x_q, x_i) = \frac{|F(x_q) \cap F(x_i)|}{|F(x_q) \cup F(x_i)|}$, where $F(\cdot)$ calculates the number of tokens in a code change. Since recent work [11, 49] has pinpointed that LLMs with ICL are more prone to be influenced by the demonstrations that are closer to the query, we arrange the demonstrations in ascending order of similarity to the queried code change. This is based on the intuition that demonstrations with higher similarity may contain more information related to the queried code change. In summary, given a query, we first select demonstrations that are similar to the query from the training set. Then based on the query and retrieved demonstrations, we construct a prompt to query ChatGPT and obtain the results.

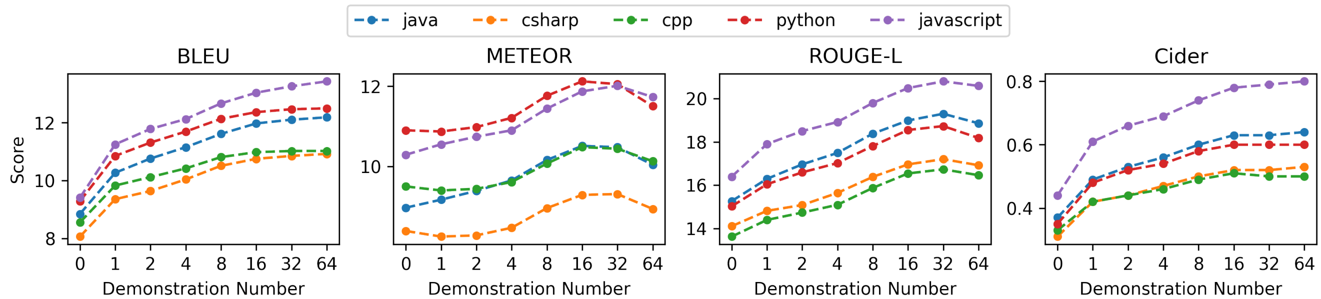

Figure 2: The results of ChatGPT on commit message generation using zero-shot, one-shot, and few-shot learning.

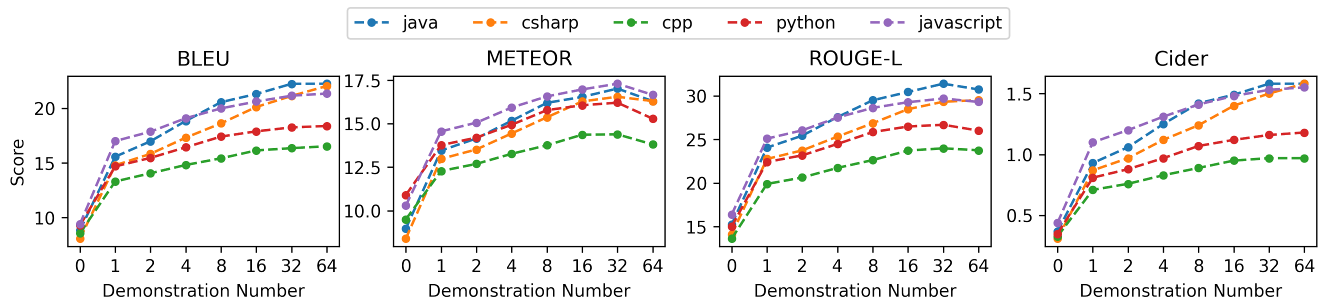

Figure 3: The results of ChatGPT with retrieval-based demonstration selection.

Table 1: The statistics of the evaluation dataset, which is a subset of MCMD.

Language	Training	Validation	Test
C++	160,948	20,000	20,141
C#	149,907	18,688	18,702
Java	160,018	19,825	20,159
Python	206,777	25,912	25,837
JavaScript	197,529	24,899	24,773

2.4 Datasets

We use the MCMD dataset [36], which is a widely-used multilingual benchmark dataset in the commit message generation task. This dataset contains five programming languages: C++, C#, Java, Python, and JavaScript. For each language, it collects commits from the top 100 most-starred repositories on GitHub. Shi et al. [35] further filter out commits with files that cannot be parsed (such as .jar, .ddl, .mp3, and .apk) to reduce noise data and build a higher-quality dataset, which is a subset of MCMD. Table 1 shows the statistics for the subset of MCMD we use in our experiment.

2.5 Evaluation Metrics

We evaluate the effectiveness of ChatGPT on four widely-used evaluation metrics in previous literature [14, 23, 35, 36, 41], including BLEU [32], METEOR [2], ROUGE-L [20], and Cider [40]. BLEU measures the precision of n-grams between the generated text and the reference texts. ROUGE-L is a recall-oriented metric that measures the longest common subsequence between the generated text and the reference texts. METEOR calculates the harmonic mean of 1-gram precision and recall of the generated text against the reference texts. Cider takes each sentence as a document and calculates the cosine similarity of its TF-IDF vector at n-gram level to obtain the similarity between the generated text and the reference texts.

2.6 Experiment Settings

We accessed and evaluated ChatGPT on the test set of the MCMD dataset using the GPT-3.5 model (`gpt-3.5-turbo-1106`) via OpenAI API [31]. Following the previous work [11, 28], we set the temperature and seed to 0 to get the deterministic output. The context window limit of ChatGPT is 16,385 tokens. Hence we cut off each demonstration to $\frac{16385}{N+1}$ tokens, where N represents the number of demonstrations.

3 STUDY RESULTS

3.1 RQ1: Effectiveness of Vanilla In-Context Learning

To investigate the effectiveness of vanilla in-context learning, we vary the number of demonstrations from 0 to 64. As shown in Figure 2, we can find that the performance of ChatGPT on all evaluation metrics increases with the number of demonstrations at first. For example, the average improvements of few-shot learning with 32 demonstrations over zero-shot learning are 35.1%, 13.0%, 24.7%, and 69.4% on BLEU, METEOR, ROUGE-L, and Cider, respectively.

Table 2: Average performance of ChatGPT and SOTA approaches on the MCMD dataset.

Method	BLEU	METEOR	ROUGE-L	Cider
Lucene [1]	16.51	10.72	19.97	0.97
NNGen [24]	17.98	11.99	23.19	1.13
CommitGen [17]	13.73	8.22	18.74	0.65
CoDiSum [43]	12.99	6.30	15.04	0.37
CoreGen [29]	19.08	12.68	25.52	1.12
CoRec [41]	17.47	11.01	23.70	1.03
RACE [35]	23.69	15.07	29.60	1.60
COME [14]	25.07	16.71	31.97	1.70
ChatGPT	19.23	16.04	27.69	1.29

However, when the number exceeds 32, BLEU and Cider tend towards stability, while METEOR and ROUGE-L begin decreasing. We attribute this to the truncation problem. As illustrated in Sec. 2.6, the length of the whole prompt will increase with the number of demonstrations, and the demonstrations might be cut off to avoid exceeding the context window limit of ChatGPT.

3.2 RQ2: Effectiveness of Demonstration Retrieval

Figure 3 shows the results of ChatGPT with retrieval-based demonstration selection. Compared with vanilla in-context learning, we can find that retrieval-based demonstration selection can significantly improve the performance of ChatGPT on commit message generation. For example, the average improvements under 32 demonstrations are 66.1%, 50.0%, 52.1%, and 121.3% on BLEU, METEOR, ROUGE-L, and Cider, respectively. Moreover, as for the impact of numbers, we can observe similar trends on ChatGPT with retrieval-based demonstration selection, all evaluation metrics first increase with the number of demonstrations. As the number further increases to 32, BLEU and Cider tend towards stability, while METEOR and ROUGE-L begin decreasing.

3.3 RQ3: Comparison with State-of-the-Art

Based on the optimal settings of ICL (i.e., 32 demonstrations and retrieval-based demonstration selection), we then compare ChatGPT with SOTA commit message generation approaches, which include IR-based approaches (i.e., Lucene [1], NNGen [24]), DL-based approaches (i.e., CommitGen [17], CoDiSum [43], CoreGen [29]), and hybrid approaches (i.e., CoRec [41], RACE [35], COME [14]), on the test set of the MCMD dataset. The experimental results are shown in Table 2. We can see that ChatGPT outperforms all the IR-based and DL-based approaches in terms of BLEU, METEOR, ROUGE-L, and Cider, and is comparable to hybrid approaches.

4 DISCUSSION

4.1 Threats to Validity

Internal validity. The training corpus of ChatGPT includes open-source projects before Sep. 2021. Thus there may be data leakage, i.e., ChatGPT may have seen the commit messages for the test cases during its pre-training. However, we observe that ChatGPT does not perform well under the zero-shot setting, which indicates that the model's output is not generated due to memorization. Another threat is the randomness of LLM inference. To mitigate this, we set the temperature and seed to 0 to generate deterministic outputs. Due to the prohibitive cost of ChatGPT API access, we did not repeat experiments multiple times.

External validity. The selection of the programming language and benchmark dataset could be a threat to the validity of our results. The results might vary depending on the programming language and size of the benchmark dataset. To mitigate these threats, we chose a popular large multilingual benchmark dataset (i.e. the MCMD dataset), which targeted five of the most popular programming languages (i.e., C++, C#, Java, Python, and JavaScript).

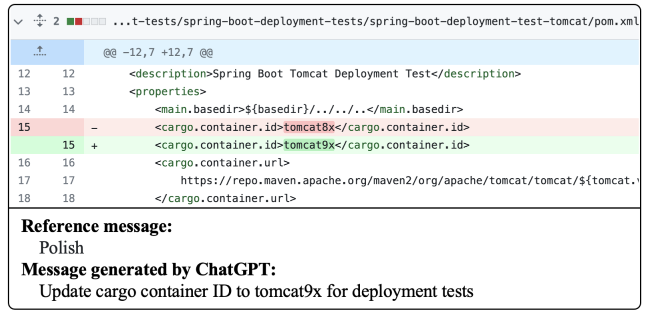

Figure 4: An example of a low-quality reference message.

4.2 Opportunities

A lot of room for improvement on existing datasets. In this study, we directly run ChatGPT on the MCMD test set without fine-tuning. From the results in Table 2, it is observed that ChatGPT's performance on commit message generation still has a lot of room for improvement on the MCMD dataset. A plausible way to improve its performance is to explore more efficient prompts. However, it is time-consuming and labor-intensive to rely on manual attempts. How to use automatic prompt engineering [44, 51] to automatically explore better prompts is worthy of further study.

Creating a new high-quality benchmark dataset. Currently, the quality of benchmark datasets widely used in commit message generation is unverified. These datasets are usually crawled from open-source projects and then subjected to simple data cleaning. According to Tian et al. [39], 44% of the commit messages from five open-source projects have quality issues. We also found some reference messages in the benchmark dataset we used are low-quality. Figure 4 shows an example of a low-quality reference message. It is observed that the reference message fails to describe what was changed in detail, while the commit message generated by ChatGPT is more informative and precisely conveys the intent of the code change. Recent works [10, 33, 45] have leveraged LLMs to generate high-quality pseudo-training sets based on their rich domain knowledge. Therefore, how to leverage LLMs to create a high-quality benchmark dataset for commit message generation is worth further exploration.

```
4  src/main/java/com/google/devtools/build/lib/exec/local/LocalSpawnRunner.java
        @@ -392,12 +392,12 @@ private SpawnResult start() throws InterruptedException, IOException {
392 392                 .profile(ProfilerTask.PROCESS_TIME, spawn.getResourceOwner().getMnemonic())) {
393 393           needCleanup = true;
394 394           Subprocess subprocess = subprocessBuilder.start();
395     -         subprocess.getOutputStream().close();
396 395           try {
    396 +           subprocess.getOutputStream().close();
397 397             subprocess.waitFor();
398 398             terminationStatus =
399 399                 new TerminationStatus(subprocess.exitValue(), subprocess.timedout());
400     -         } catch (InterruptedException e) {
    400 +         } catch (InterruptedException | IOException e) {
401 401             subprocess.destroyAndWait();
402 402             throw e;
403 403           }
```

Reference message:
Fix possible subprocess leak.
Message generated by COME:
fix a race condition in localspawnrunner.
(BLEU: 19.21, METEOR: 12.03, ROUGE-L: 34.37)
Message generated by ChatGPT:
Handle IOException when closing subprocess output stream and waiting for subprocess termination in LocalSpawnRunner.
(BLEU: 8.36, METEOR: 9.82, ROUGE-L: 21.98)

Figure 5: An example of traditional metrics not suitable for evaluating the quality of messages generated by ChatGPT.

Designing new metrics to evaluate commit message generation approaches. Traditional metrics (e.g., BLEU) provide a quick assessment by quantifying the overlap of words or characters between the generated and the reference messages. However, these metrics often fail to capture semantic quality like informativeness or usefulness, and their reliability can be further undermined if the reference messages are of poor quality. As shown in Figure 5, it is observed that although the message generated by COME outperforms the one generated by ChatGPT in BLEU, METEOR, and ROUGE-L, the semantic of "subprocess" is missing in the message generated by COME. However, the message generated by ChatGPT covers the semantics of "subprocess" and even provides more information than the reference message. This is strong evidence that traditional metrics are no longer suitable for evaluating the quality of the comments generated by ChatGPT and highlights the need for designing new metrics to evaluate commit message generation approaches from a semantic perspective.

5 CONCLUSION

In this paper, we conduct a preliminary evaluation of ChatGPT with in-context learning on commit message generation. We compare ChatGPT with several SOTA approaches on a large multilingual dataset. We find that a carefully-designed demonstration can lead to substantial improvements for ChatGPT on commit message generation. Furthermore, ChatGPT outperforms all the IR-based and DL-based approaches in terms of BLEU, METEOR, ROUGE-L, and Cider, and is comparable to hybrid approaches. Based on our findings, we outline three open challenges and opportunities for applying ChatGPT to commit message generation. We hope our work will be helpful for future research on ChatGPT-based commit message generation.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. This work was supported by Ant Group.

REFERENCES

[1] Apache. 2011. Apache Lucene. https://lucene.apache.org

[2] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[4] Raymond PL Buse and Westley R Weimer. 2010. Automatically documenting program changes. In *Proceedings of the 25th IEEE/ACM international conference on automated software engineering*. 33–42.

[5] Luis Fernando Cortés-Coy, Mario Linares-Vásquez, Jairo Aponte, and Denys Poshyvanyk. 2014. On automatically generating commit messages via summarization of source code changes. In *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. IEEE, 275–284.

[6] Jinhao Dong, Yiling Lou, Qihao Zhu, Zeyu Sun, Zhilin Li, Wenjie Zhang, and Dan Hao. 2022. FIRA: fine-grained graph-based code change representation for automated commit message generation. In *Proceedings of the 44th International Conference on Software Engineering*. 970–981.

[7] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[8] Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N Nguyen. 2013. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 422–431.

[9] Aleksandra Eliseeva, Yaroslav Sokolov, Egor Bogomolov, Yaroslav Golubev, Danny Dig, and Timofey Bryksin. 2023. From commit message generation to history-aware commit message completion. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 723–735.

[10] Jiahui Gao, Renjie Pi, Lin Yong, Hang Xu, Jiacheng Ye, Zhiyong Wu, Weizhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. 2023. Self-guided noise-free data generation for efficient zero-shot learning. In *International Conference on Learning Representations*.

[11] Shuzheng Gao, Xin-Cheng Wen, Cuiyun Gao, Wenxuan Wang, Hongyu Zhang, and Michael R Lyu. 2023. What Makes Good In-Context Demonstrations for Code Intelligence Tasks with LLMs?. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE Computer Society, 761–773.

[12] Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Ge Li, Zhi Jin, Xiaoguang Mao, and Xiangke Liao. 2024. Large Language Models are Few-Shot Summarizers: Multi-Intent Comment Generation via In-Context Learning. (2024).

[13] Yaroslav Golubev, Viktor Poletansky, Nikita Povarov, and Timofey Bryksin. 2021. Multi-threshold token-based code clone detection. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 496–500.

[14] Yichen He, Liran Wang, Kaiyi Wang, Yupeng Zhang, Hang Zhang, and Zhoujun Li. 2023. COME: Commit Message Generation with Modification Embedding. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 792–803.

[15] Yuan Huang, Nan Jia, Hao-Jie Zhou, Xiang-Ping Chen, Zi-Bin Zheng, and Ming-Dong Tang. 2020. Learning human-written commit messages to document code changes. *Journal of Computer Science and Technology* 35 (2020), 1258–1277.

[16] 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)*. IEEE, 414–423.

[17] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 135–146.

[18] Toshihiro Kamiya, Shinji Kusumoto, and Katsuro Inoue. 2002. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *IEEE transactions on software engineering* 28, 7 (2002), 654–670.

[19] Jiawei Li and Iftekhar Ahmed. 2023. Commit message matters: Investigating impact and evolution of commit message quality. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 806–817.

[20] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[21] Mario Linares-Vásquez, Luis Fernando Cortés-Coy, Jairo Aponte, and Denys Poshyvanyk. 2015. Changescribe: A tool for automatically generating commit messages. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 709–712.

[22] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What Makes Good In-Context Examples for GPT-3?. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on*

Knowledge Extraction and Integration for Deep Learning Architectures. 100–114.
[23] Shangqing Liu, Cuiyun Gao, Sen Chen, Lun Yiu Nie, and Yang Liu. 2020. ATOM: Commit message generation based on abstract syntax tree and hybrid ranking. *IEEE Transactions on Software Engineering* 48, 5 (2020), 1800–1817.
[24] 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*. 373–384.
[25] Cristina V Lopes, Vanessa I Klotzman, Iris Ma, and Iftekar Ahmed. 2024. Commit Messages in the Age of Large Language Models. *arXiv preprint arXiv:2401.17622* (2024).
[26] Walid Maalej and Hans-Jörg Happel. 2010. Can development work describe itself?. In *2010 7th IEEE working conference on mining software repositories (MSR 2010)*. IEEE, 191–200.
[27] Mockus and Votta. 2000. Identifying reasons for software changes using historic databases. In *Proceedings 2000 International Conference on Software Maintenance*. IEEE, 120–130.
[28] Noor Nashid, Mifta Sintaha, and Ali Mesbah. 2023. Retrieval-based prompt selection for code-related few-shot learning. In *Proceedings of the 45th International Conference on Software Engineering (ICSE'23)*.
[29] Lun Yiu Nie, Cuiyun Gao, Zhicong Zhong, Wai Lam, Yang Liu, and Zenglin Xu. 2021. Coregen: Contextualized code representation learning for commit message generation. *Neurocomputing* 459 (2021), 97–107.
[30] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. 2013. Using of Jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, Vol. 1. 380–384.
[31] OpenAI. 2023. Introducing ChatGPT. https://openai.com/blog/chatgpt
[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.
[33] WANG Ruida, Wangchunshu Zhou, and Mrinmaya Sachan. 2023. Let's Synthesize Step by Step: Iterative Dataset Synthesis with Large Language Models by Extrapolating Errors from Small Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
[34] 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 *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, 103–112.
[35] Ensheng Shi, Yanlin Wang, Wei Tao, Lun Du, Hongyu Zhang, Shi Han, Dongmei Zhang, and Hongbin Sun. 2022. RACE: Retrieval-augmented Commit Message Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 5520–5530.
[36] Wei Tao, Yanlin Wang, Ensheng Shi, Lun Du, Shi Han, Hongyu Zhang, Dongmei Zhang, and Wenqiang Zhang. 2022. A large-scale empirical study of commit message generation: models, datasets and evaluation. *Empirical Software Engineering* 27, 7 (2022), 198.
[37] Wei Tao, Yucheng Zhou, Yanlin Wang, Hongyu Zhang, Haofen Wang, and Wenqiang Zhang. 2024. KADEL: Knowledge-Aware Denoising Learning for Commit Message Generation. *ACM Transactions on Software Engineering and Methodology* (2024).
[38] Yida Tao, Yingnong Dang, Tao Xie, Dongmei Zhang, and Sunghun Kim. 2012. How do software engineers understand code changes? An exploratory study in industry. In *Proceedings of the ACM SIGSOFT 20th International symposium on the foundations of software engineering*. 1–11.
[39] Yingchen Tian, Yuxia Zhang, Klaas-Jan Stol, Lin Jiang, and Hui Liu. 2022. What makes a good commit message?. In *Proceedings of the 44th International Conference on Software Engineering*. 2389–2401.
[40] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4566–4575.
[41] Haoye Wang, Xin Xia, David Lo, Qiang He, Xinyu Wang, and John Grundy. 2021. Context-aware retrieval-based deep commit message generation. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 4 (2021), 1–30.
[42] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 8696–8708.
[43] Shengbin Xu, Yuan Yao, Feng Xu, Tianxiao Gu, Hanghang Tong, and Jian Lu. 2019. Commit message generation for source code changes. In *28th International Joint Conference on Artificial Intelligence, IJCAI 2019*. International Joint Conferences on Artificial Intelligence, 3975–3981.
[44] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409* (2023).
[45] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. ZeroGen: Efficient Zero-shot Learning via Dataset Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 11653–11669.
[46] Aiping Zhang, Liming Fang, Chunpeng Ge, Piji Li, and Zhe Liu. 2023. Efficient transformer with code token learner for code clone detection. *Journal of Systems and Software* 197 (2023), 111557.
[47] Linghao Zhang, Jingshu Zhao, Chong Wang, and Peng Liang. 2024. Using Large Language Models for Commit Message Generation: A Preliminary Study. *arXiv preprint arXiv:2401.05926* (2024).
[48] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).
[49] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*. PMLR, 12697–12706.
[50] Zibin Zheng, Kaiwen Ning, Yanlin Wang, Jingwen Zhang, Dewu Zheng, Mingxi Ye, and Jiachi Chen. 2023. A Survey of Large Language Models for Code: Evolution, Benchmarking, and Future Trends. *arXiv preprint arXiv:2311.10372* (2023).
[51] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910* (2022).