Commit Classification into Maintenance Activities Using In-Context Learning Capabilities of Large Language Models

Yasin Sazid[1], Sharmista Kuri[1], Kazi Solaiman Ahmed[2] and Abdus Satter[1]
[1]*Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh*
[2]*Computer Science Department, University of New Mexico, Albuquerque, New Mexico, U.S.A.*

Keywords: Commit Classification, Commit Message, Maintenance Activity, Large Language Models, GPT, In-Context Learning.

Abstract: Classifying software changes, i.e., commits into maintenance activities enables improved decision-making in software maintenance, thereby decreasing maintenance costs. Commonly, researchers have tried commit classification using keyword-based analysis of commit messages. Source code changes and density data have also been used for this purpose. Recent works have leveraged contextual semantic analysis of commit messages using pre-trained language models. But these approaches mostly depend on training data, making their ability to generalize a matter of concern. In this study, we explore the possibility of using in-context learning capabilities of large language models in commit classification. In-context learning does not require training data, making our approach less prone to data overfitting and more generalized. Experimental results using GPT-3 achieves a highest accuracy of 75.7% and kappa of 61.7%. It is similar to performances of other baseline models except one, highlighting the applicability of in-context learning in commit classification.

1 INTRODUCTION

Software maintenance constitutes a significant portion of the overall costs in software development. To increase cost-effectiveness, it is imperative to understand the different maintenance activities involved with software development (Swanson, 1976). Categorization of these maintenance activities enables decision-making on resource allocation, choice of technology, and management of technical debt (Ghadhab et al., 2021), making it easier to manage costs. To realize this benefit, many researchers have tried to profile software projects in terms of maintenance activities (Swanson, 1976) (Mockus and Votta, 2000) (Levin and Yehudai, 2016). The very first task in maintenance activity profiling is commit classification. As commits keep track of technical changes in a software, classifying them into maintenance activities helps better understand and manage software maintenance, thereby improving cost-effectiveness.

Different approaches have been proposed for commit classification. The most common approach is analyzing commit messages. Multiple studies have employed keyword-based analysis of commit messages (Hindle et al., 2009) (Levin and Yehudai, 2017) (Mariano et al., 2021). Aside from commit messages, studies also considered other data sources, such as source code changes (Levin and Yehudai, 2017) (Meqdadi et al., 2019) and code density (Hönel et al., 2020). Other studies have tried contextual semantic analysis of commit messages by fine-tuning pre-trained language models (Ghadhab et al., 2021) (Zafar et al., 2019). However, such approaches mostly depend on training machine learning classifiers with commit data, thus making them dependent on the quality of training data.

In this study, we propose the use of in-context learning capabilities of large language models (LLMs) in the context of commit classification. As in-context learning does not require training data, it can be an excellent way to generalize commit classification across commit data from different sources. We previously achieved encouraging results in detecting and classifying user interface (UI) dark pattern texts using in-context learning (Sazid et al., 2023). We use the same approach in this study to apply in-context learning in the context of commit message classification. In this approach, we first synthesize definitions of maintenance activity categories from the existing literature. These category definitions are then used to engineer prompts for the large language model, along with zero, one, or two examples per category.

Sazid, Y., Kuri, S., Ahmed, K. and Satter, A.
Commit Classification into Maintenance Activities Using In-Context Learning Capabilities of Large Language Models.
DOI: 10.5220/0012686700003687

In *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2024)*, pages 506-512
ISBN: 978-989-758-696-5; ISSN: 2184-4895

Experimental results using GPT-3 present encouraging signs about the applicability of in-context learning in the context of commit classification. We achieve a highest accuracy of 75.7% and kappa of 61.7%, which is similar to other baseline approaches except one. Most importantly, our approach requires no training data. It only uses semantics of category definitions to classify commit message texts. Thus, it is less prone to data overfitting and offers increased generalization over other approaches.

2 RELATED WORK

Commit classification is not a new field of research, but recent years have seen a growing number of studies on this topic (Heričko and Šumak, 2023). As the necessity to understand and organize software changes keeps rising, researchers have tried to automate the commit classification process using rule-based models (Amit and Feitelson, 2021) (Hassan, 2008) (Mauczka et al., 2012), supervised (Ghadhab et al., 2021) (Hindle et al., 2009) (Levin and Yehudai, 2017) (Hönel et al., 2020) (Zafar et al., 2019) and semi-supervised (Fu et al., 2015) (Gharbi et al., 2019) machine learning.

Hindle et al. used machine learning to classify large changes into five maintenance categories - corrective, adaptive, perfective, feature addition, and non-functional changes (Hindle et al., 2009). They used the commit message, author, and modified modules data in their work. This trend of analyzing commit messages can be seen in other works as well. Fu et al. and Yan et al. used topic modelling on commit messages (Fu et al., 2015) (Yan et al., 2016). But the most common technique in analyzing commit messages is word frequency-based analysis (Heričko and Šumak, 2023). Levin et al. used such an approach to extract keywords from commit messages (Levin and Yehudai, 2017). They used the presence of keywords and source code changes (number of statements, methods, files changed etc.) to distinguish between commits belonging to three maintenance activities - corrective, adaptive, and perfective. Hönel et al. extended the work of Levin et al. by incorporating source code density with keywords and code changes to improve the classification accuracy (Hönel et al., 2020) (Levin and Yehudai, 2017). However, keyword-based techniques do not recognize the contextual relationship between words. As a result, it is necessary to train a context-aware model for commit classification.

Multiple studies used pre-trained language models for contextual analysis of commit messages. Ghadhab et al. used fine-grained code changes with a pre-trained language model, BERT (Bidirectional Encoder Representations from Transformers), to augment commit classification (Ghadhab et al., 2021). Sarwar et al. and Trautsch et al. also used contextual semantic analysis of commit messages (Sarwar et al., 2020) (Trautsch et al., 2023). Zafar et al. presented a set of rules for semantic analysis of commit messages and trained a context-aware deep learning model by fine-tuning BERT for bug-fix commit message classification (Zafar et al., 2019). These applications of pre-trained language models generally focus on fine-tuning models with commit message data. As a result, they are only as generalized as the source of training data.

Using the semantics of category definitions of maintenance activities can provide better generalization capabilities for language models. Large Language Models (LLMs) like GPT (Generative Pre-trained Transformer) have in-context learning capabilities that can be useful in this regard. This capability enables LLMs to perform tasks by conditioning on only a few examples (Min et al., 2022). What separates our work from existing works is that we leverage this capability of LLMs for commit classification instead of fine-tuning the pre-trained models with commit message data.

3 BACKGROUND

We start this section with an overview of maintenance activity categories considered in this work. Then, we introduce in-context learning and GPT, a large language model that has been illustrating revolutionary capabilities in different natural language processing (NLP) tasks, especially text classification. Finally, we explain statistical methods used in this study to evaluate classification performance.

3.1 Maintenance Activity Categories

Swanson proposed three maintenance activity categories - ‘Corrective’, ‘Perfective’, and ‘Adaptive’ (Swanson, 1976). These categories are briefly explained in this subsection.

3.1.1 Corrective

Corrective maintenance involves identifying and fixing issues like bugs and faults in software to ensure it behaves as intended. It addresses problems in processing or performance, caused by errors in the application software, hardware, or system software. It is

carried out in response to failures and includes tasks like bug fixing, resolving processing issues, addressing performance problems, and correcting implementation failures.

3.1.2 Perfective

Perfective maintenance involves improving a software, even if it is already built and documented well, without any implementation problems. The goal is to make the software easier to modify during corrective or adaptive maintenance, not necessarily to reduce program failures or handle environmental changes better. This type of maintenance focuses on getting rid of inefficiencies, enhancing performance, and improving maintainability, ultimately striving for a more perfect design.

3.1.3 Adaptive

Adaptive maintenance involves adjusting software in response to changes in the data and processing environments. For instance, changes in data might involve modifying classification codes or restructuring a database, while changes in processing could result from new hardware or operating system installations, requiring updates to existing programs. It's crucial to anticipate these changes for timely and effective adaptive maintenance.

3.2 In-Context Learning

In-context learning is a prompt engineering strategy for large language models that do not require traditional sense of training in machine learning. Instead of training or fine-tuning models with large datasets, only a few examples are provided within the prompt. As a result, models can learn tasks using inference only, without updating underlying parameters (Min et al., 2021).

3.3 Generative Pre-Trained Transformer (GPT)

GPT (Generative Pre-trained Transformer) is a series of large language models developed by OpenAI[1]. They are based on a transformer architecture, which is a type of deep learning model that uses attention mechanisms to capture relationships between words in a sentence. GPT excels in a variety of natural language processing tasks, such as question answering, translation, summarizing, classification, and text parsing (Chiu et al., 2021).

[1]https://openai.com

3.4 Statistical Methods

We evaluate our classification performance using common statistical measures like accuracy, precision, recall and kappa. Short descriptions of the statistical methods required in this study are provided below.

- True Positive (TP) - Number of instances that were correctly classified as belonging to a class.
- False Positive (FP) - Number of instances that were incorrectly classified as belonging to a class when they actually belong to a different class.
- True Negative (TN) - Number of instances that were correctly classified as not belonging to a class.
- False Negative (FP) - Number of instances that were incorrectly classified as not belonging to a class when they actually belong to that class.
- Precision - Ratio of correctly classified instances of a class over all instances that were classified as belonging to that class.

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

- Recall - Ratio of correctly classified instances of a class over all instances of that class.

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

- Accuracy - Ratio of correctly classified instances over all instances.

$$Accuracy = \frac{\text{\# of correctly classified instances}}{\text{\# of all instances}} \tag{3}$$

- Kappa - Cohen's kappa is a metric used to account for uneven distribution of classification categories or classes. It considers the possibility of the agreement occurring by chance, therefore providing a more reliable metric than accuracy.

4 METHODOLOGY

Our main objective in this study is to measure the applicability of in-context learning in the context of commit classification. Previously, we executed such an investigation in the context of automated detection of dark pattern texts (Sazid et al., 2023). We follow the same approach of applying in-context learning in this new context of commit classification. We begin by synthesizing comprehensive definitions of the three maintenance activity categories considered in this study. This information is then utilized to engineer prompts for large language models. We choose

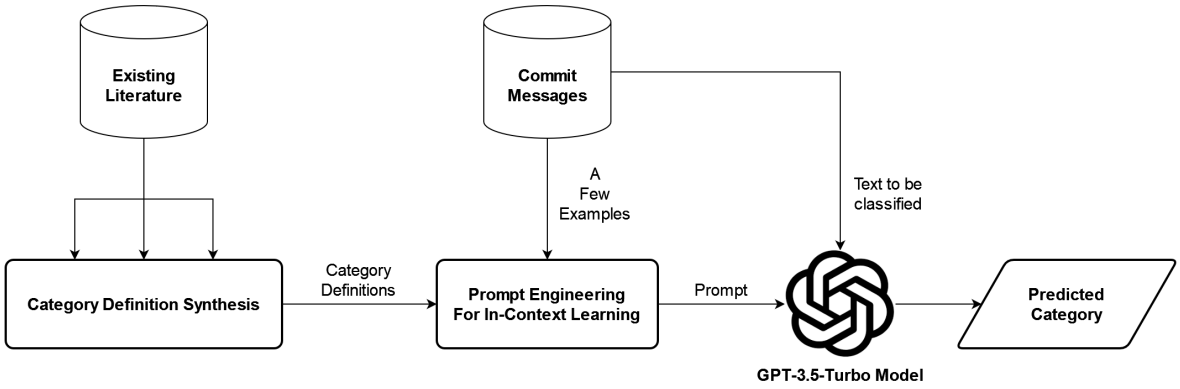

Figure 1: Overview of Commit Classification Approach Using In-Context Learning.

GPT-3 as the large language model to be used in this study because of its encouraging performance in terms of in-context learning capabilities. Figure 1 depicts an overview of the commit classification approach used in this study.

4.1 Dataset of Commit Messages

We utilize the commit dataset provided by Levin et al., which comprises 1,151 commit data from 11 Java projects (Levin and Yehudai, 2017). Commits in this dataset are categorized as 'Adaptive', 'Corrective', or 'Perfective'. The distribution of commits is as follows: 22% 'Adaptive', 43% 'Corrective', and 35% 'Perfective'. Aside from the commit message, the dataset also includes source code changes and keywords for each commit. But we do not use these other features in our work. 1,145 of the 1,151 commits are utilized solely to validate GPT-3 classification models. The remaining 6 commits illustrated in Table 1 are utilized for training or validation based on GPT-3 prompting techniques.

4.2 Category Definition Synthesis

We must first synthesize the definitions of the maintenance activity categories. We want to use this as contextual information in our classification process. As a result, the meaning and phrasing of the category definitions will have a significant influence on the classification outcomes. We synthesize relevant phrasings and characteristics from the literature (Swanson, 1976) (Mockus and Votta, 2000) (Levin and Yehudai, 2017) (Heričko and Šumak, 2023) to provide a detailed and comprehensive definition for each maintenance activity category. We enumerate the different phrasings to generate each category definition. The resulting definitions are listed in Table 1.

4.3 Classification of Commit Messages

Similar to our previous work on in-context learning, we utilize the 'GPT for Sheets™ and Docs™' extension available in 'Google Sheets' to classify all the commit message texts in the dataset. We employ the 'GPT()' function provided by this extension. It sends a prompt to GPT and returns the result. We select the 'gpt-3.5-turbo' model for this study. As commit message classification necessitates precise results, we configure the 'temperature' parameter of the model to 0 so that it prioritizes accuracy over creative results. After configuration, We send engineered prompts to the model appending a single piece of commit message text to classify that text. We call the 'GPT()' function for all the commit message texts in the dataset following this approach. The 'GPT()' function call operates in a completely new session each time, ensuring that GPT's memory of the previous context does not influence the outcomes.

4.4 Prompt Engineering

Prompt engineering is carried out to apply in-context learning for the following prompting techniques - zero-shot, one-shot, and few-shot. Procedures for prompt engineering followed in this study are explained in this subsection.

4.4.1 Zero-Shot Prompting

In zero-shot prompting, no example for any of the classification categories is provided to the GPT-3 classification model. A template for zero-shot learning used in our study is given below -

Prompt:

Classify the commit message into 'adaptive', 'perfective' or 'corrective' maintenance activities.

<Definition of 'Corrective' Category>

Table 1: Definitions and Examples of Classification Categories.

Classification Category	Definition	Examples
Corrective	'Corrective' means 'changes made to fix functional and non-functional faults, failures, and errors'.	1. Percolator response now always returns the- 'matches' key.–Closes -4881- 2. Wraps DoOnEach in a SafeObserver–This commit leverages the SafeObserver facility to get the desired-behavior in the face of exceptions. Specifically, if any of the-operations performed within the doOnEach handler raises an exception,-that exception will propagate through the observable chain.-
Perfective	'Perfective' means 'changes made to improve software quality attributes such as processing efficiency, performance, and maintainability'.	1. Fixed typos in test.- 2. HBASE-6667 TestCatalogJanitor occasionally fails;- PATCH THAT ADDS DEBUG AROUND FAILING TEST–git-svn-id: https://svn.apache.org/repos/asf/hbase/trunk@1379682 13f79535-47bb-0310-9956-ffa450edef68-
Adaptive	'Adaptive' means 'adding or introducing new features into the system', and 'changes made to adapt to changes in the data and processing environment'.	1. plugins: tooltip for plugins with newer version- (IDEA-75998)– 2. Create from usage: Create constructor parameter- by reference in delegation specifier -KT-6601 Fixed–

<Definition of 'Perfective' Category>
<Definition of 'Adaptive' Category>
Now I will give you one commit message, and you will have to say which category it belongs to. You must reply with only one letter - 'c' for 'corrective', 'p' for 'perfective' or 'a' for 'adaptive'.
<Text to be classified>
GPT-3 Response: <Predicted Category>

4.4.2 One-Shot Prompting

In one-shot prompting, one example per category is provided with the definitions. Each <Definition of Category X> is followed by <Example 1 of Category X>. The first example for each category shown in Table 1 is used in one-shot prompting. Example text selection is carried out randomly and they are removed from the validation dataset.

4.4.3 Few-Shot Prompting

We use two examples per category in few-shot prompting. Each <Definition of Category X> is followed by <Example 1 of Category X> and <Example 2 of Category X>. The example texts used in one-shot are reused in few-shot, while the other three example texts are randomly selected. All the example texts used in prompting are shown in Table 1, which are removed from the validation dataset.

5 EVALUATION

We use accuracy, precision, recall and kappa to assess the performance of our approach for commit classification. Overall classification performance across prompting techniques are listed in Table 2.

5.1 Result Analysis

Among the three prompting techniques used in this study, zero-shot prompting produces the best results with an overall accuracy of 75.7% and a kappa of 61.7%. It accurately classifies 871 out of 1,151 commit messages. Zero-shot prompting works best for the 'Corrective' category with a precision of 88% and a recall of 80%. One-shot prompting sees a decrease in performance with an overall accuracy of 70% and a kappa of 52.5%. It accurately classifies 804 out of 1,148 commit messages. Like zero-shot, one-shot prompting also works best for the 'Corrective' category with a precision of 86% and a recall of 73%, even though both see a decrease compared to zero-shot. Few-shot prompting further decreases the overall accuracy to 65.6% and the kappa to 45.6%. 751 out of 1,145 commit messages were accurately classified using few-shot prompting. The 'Perfective' category has high recall scores and low precision scores across all three prompting techniques. The 'Adaptive' category illustrates totally opposite results. It has high precision scores and low recall scores across prompting techniques.

Figure 2 depicts a steady decrease in accuracy and kappa as the number of examples is increased in the prompt. This can be explained by the fact that commit message texts are diverse in representation. Developers from different companies, projects, or geographic locations may write commit messages in different ways. As a result, providing a few examples makes the classifier more biased rather than generalized. Thus, in-context learning for commit classification works best when relying only on the semantics of the category definitions. More accurate and detailed definitions of categories might further facilitate commit classification using in-context learning.

Table 2: Experimental Results Using GPT-3.

Category	Zero-Shot				One-Shot				Few-Shot			
	Precision	Recall	Accuracy	Kappa	Precision	Recall	Accuracy	Kappa	Precision	Recall	Accuracy	Kappa
Corrective	88%	80%	75.7%	61.7%	86%	73%	70%	52.5%	91%	62%	65.6%	45.6%
Perfective	63%	83%			57%	87%			51%	94%		
Adaptive	81%	55%			83%	36%			90%	25%		

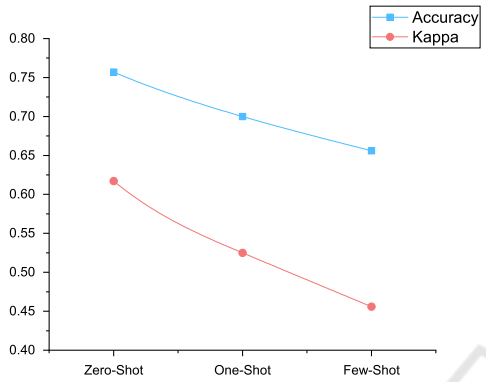

Figure 2: Performance Across Prompting Techniques.

5.2 Result Discussion

Table 3 compares our best performance, achieved using zero-shot prompting, with some baseline multi-class commit classification models trained and validated on the same dataset used in this study. It is easily visible that our performance is similar to the other models except the one using the LogitBoost classifier (Hönel et al., 2020). Even though our approach fails to beat the model using LogitBoost, it is important to consider a significant difference between all of these models and our approach. These models learned from training data making them prone to data overfitting whereas we use in-context learning. Our best performance is achieved without any training, using only the semantics of category definitions. As a result, our approach is less prone to overfitting and more generalized compared to other models.

As zero-shot is our best-performing model, using more examples can not improve the performance of our approach. However, there are two feasible directions for improvement. Firstly, more detailed and accurate category definitions in prompts can potentially enhance performance. Secondly, integrating our approach with other non-ML methods, such as keyword analysis, source code change analysis, or topic modelling, could further improve performance. Such an amalgamation can leverage the generalization capabilities of in-context learning while also benefiting from the classification capabilities of other methods.

Table 3: Performance Comparison With Baseline Models.

Commit Classification Model	Year	Accuracy	Kappa
Multi-class classification based on source code changes and keywords extracted from commit messages using LogitBoost (Hönel et al., 2020)	2020	85%	78%
Multi-class classification based on source code changes and keywords extracted from commit messages using Random Forest (Levin and Yehudai, 2017)	2017	73.6%	58.9%
Multi-class classification based on quantitative metrics and keywords extracted from commit messages using Random Forest (Mariano et al., 2021)	2021	75.7%	62.4%
Multi-class classification based on commit messages using In-Context Learning (proposed)	2023	75.7%	61.7%

6 THREATS TO VALIDITY

Validation with a single dataset poses an external validity threat. Classification performance may differ in other commit datasets. However, our approach is generalized in terms of training data. As we do not use training data, the approach can be as generalized as the category definitions used in prompt engineering for large language models. As a result, we believe this approach can be useful in classifying commit messages from other sources as well. Random example selection for prompt engineering poses a threat to the internal validity of the study. However, in zero-shot prompting, no example was used in the prompt. Thus, the performance of zero-shot prompting, which is the best-performing model in this study, does not suffer from this issue.

7 CONCLUSION

In this study, we investigate the applicability of in-context learning in the context of commit classification. Our approach synthesizes definitions of maintenance activity categories from the existing literature, which are used as contextual information for large language models to classify commit messages. Experimental results using GPT-3 show encouraging

performance in zero-shot compared to other baseline models. In-context learning decreases the risk of data overfitting as no training data is used. Thus, our commit classification approach is as generalized as the category definitions used in prompt engineering. In the future, we plan to combine this approach with other commit classification approaches to further improve the classification performance.

REFERENCES

Amit, I. and Feitelson, D. G. (2021). Corrective commit probability: a measure of the effort invested in bug fixing. *Software Quality Journal*, 29(4):817–861.

Chiu, K.-L., Collins, A., and Alexander, R. (2021). Detecting hate speech with gpt-3. *arXiv preprint arXiv:2103.12407*.

Fu, Y., Yan, M., Zhang, X., Xu, L., Yang, D., and Kymer, J. D. (2015). Automated classification of software change messages by semi-supervised latent dirichlet allocation. *Information and Software Technology*, 57:369–377.

Ghadhab, L., Jenhani, I., Mkaouer, M. W., and Messaoud, M. B. (2021). Augmenting commit classification by using fine-grained source code changes and a pre-trained deep neural language model. *Information and Software Technology*, 135:106566.

Gharbi, S., Mkaouer, M. W., Jenhani, I., and Messaoud, M. B. (2019). On the classification of software change messages using multi-label active learning. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1760–1767.

Hassan, A. E. (2008). Automated classification of change messages in open source projects. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 837–841.

Heričko, T. and Šumak, B. (2023). Commit classification into software maintenance activities: A systematic literature review. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1646–1651. IEEE.

Hindle, A., German, D. M., Godfrey, M. W., and Holt, R. C. (2009). Automatic classication of large changes into maintenance categories. In *2009 IEEE 17th International Conference on Program Comprehension*, pages 30–39. IEEE.

Hönel, S., Ericsson, M., Löwe, W., and Wingkvist, A. (2020). Using source code density to improve the accuracy of automatic commit classification into maintenance activities. *Journal of Systems and Software*, 168:110673.

Levin, S. and Yehudai, A. (2016). Using temporal and semantic developer-level information to predict maintenance activity profiles. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 463–467. IEEE.

Levin, S. and Yehudai, A. (2017). Boosting automatic commit classification into maintenance activities by utilizing source code changes. In *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 97–106.

Mariano, R. V., dos Santos, G. E., and Brandão, W. C. (2021). Improve classification of commits maintenance activities with quantitative changes in source code. In *ICEIS (2)*, pages 19–29.

Mauczka, A., Huber, M., Schanes, C., Schramm, W., Bernhart, M., and Grechenig, T. (2012). Tracing your maintenance work–a cross-project validation of an automated classification dictionary for commit messages. In *Fundamental Approaches to Software Engineering: 15th International Conference, FASE 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24-April 1, 2012. Proceedings 15*, pages 301–315. Springer.

Meqdadi, O., Alhindawi, N., Alsakran, J., Saifan, A., and Migdadi, H. (2019). Mining software repositories for adaptive change commits using machine learning techniques. *Information and Software Technology*, 109:80–91.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. (2021). Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Mockus and Votta (2000). Identifying reasons for software changes using historic databases. In *Proceedings 2000 International Conference on Software Maintenance*, pages 120–130. IEEE.

Sarwar, M. U., Zafar, S., Mkaouer, M. W., Walia, G. S., and Malik, M. Z. (2020). Multi-label classification of commit messages using transfer learning. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 37–42. IEEE.

Sazid, Y., Fuad, M. M. N., and Sakib, K. (2023). Automated detection of dark patterns using in-context learning capabilities of gpt-3. In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, pages 569–573. IEEE.

Swanson, E. B. (1976). The dimensions of maintenance. In *Proceedings of the 2nd international conference on Software engineering*, pages 492–497.

Trautsch, A., Erbel, J., Herbold, S., and Grabowski, J. (2023). What really changes when developers intend to improve their source code: a commit-level study of static metric value and static analysis warning changes. *Empirical Software Engineering*, 28(2):30.

Yan, M., Fu, Y., Zhang, X., Yang, D., Xu, L., and Kymer, J. D. (2016). Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project. *Journal of Systems and Software*, 113:296–308.

Zafar, S., Malik, M. Z., and Walia, G. S. (2019). Towards standardizing and improving classification of bug-fix commits. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6. IEEE.