Leveraging GPT-like LLMs to Automate Issue Labeling

Giuseppe Colavito
giuseppe.colavito@uniba.it
University of Bari
Italy

Filippo Lanubile
filippo.lanubile@uniba.it
University of Bari
Italy

Nicole Novielli
nicole.novielli@uniba.it
University of Bari
Italy

Luigi Quaranta
luigi.quaranta@uniba.it
University of Bari
Italy

ABSTRACT

Issue labeling is a crucial task for the effective management of software projects. To date, several approaches have been put forth for the automatic assignment of labels to issue reports. In particular, supervised approaches based on the fine-tuning of BERT-like language models have been proposed, achieving state-of-the-art performance. More recently, decoder-only models such as GPT have become prominent in SE research due to their surprising capabilities to achieve state-of-the-art performance even for tasks they have not been trained for. To the best of our knowledge, GPT-like models have not been applied yet to the problem of issue classification, despite the promising results achieved for many other software engineering tasks. In this paper, we investigate to what extent we can leverage GPT-like LLMs to automate the issue labeling task. Our results demonstrate the ability of GPT-like models to correctly classify issue reports in the absence of labeled data that would be required to fine-tune BERT-like LLMs.

CCS CONCEPTS

• **Software and its engineering → Documentation**; **Software evolution**; **Maintaining software**; • **Information systems → Clustering and classification**;

KEYWORDS

LLM, Issue Labeling, GPT, Software Maintenance and Evolution, Labeling Unstructured Data

ACM Reference Format:
Giuseppe Colavito, Filippo Lanubile, Nicole Novielli, and Luigi Quaranta. 2024. Leveraging GPT-like LLMs to Automate Issue Labeling. In *Proceedings of The 21st International Conference on Mining Software Repositories (MSR 2024).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Large Language Models (LLMs) are increasingly receiving the attention of researchers in the software engineering field, where they are currently being applied to a variety of tasks such as testing, code generation, or code summarization [23, 34]. To date, traditional software engineering (SE) tasks concerned with the analysis of natural language have been mainly addressed using encoder-only LLMs, such as BERT [21] and its variants [45, 61, 88], also including consideration of SE-specific optimizations such as CodeBERT [25] or BERTOverflow [64]. Also encoder-decoder LLMs have been explored, such as T5 [58] and CodeT5 [75]. More recently (i.e., starting from 2023) decoder-only models, such as GPT [50–52, 57] or LLaMA [65, 66], have become prominent in SE research [34].

Among the SE tasks that are traditionally addressed as natural language processing (NLP) problems, issue report classification based on textual content is a task that has been largely investigated, being regarded as critical for prioritizing and coordinating work[4, 55]. Early work on issue classification focused on distinguishing between bugs and feature requests [4], using traditional machine-learning techniques and text-based features. Herzig et al. [32] aimed at distinguishing between six categories: bug, feature request, improvement request, documentation request, and others. Kallis et al. [39] proposed Ticket Tagger, a tool that leverages fastText [6] to represent and classify the textual content of issue titles and bodies in three categories: bug, feature request, and question.

More recently, supervised approaches based on BERT-like language models have been proposed by Izadi et al. [35, 36], achieving state-of-the-art performance. However, evaluation results highlighted several challenges in correctly classifying minority classes. This problem is particularly relevant when trying to implement fine-grained labeling schemes, which typically lead to limited data available for each class. On the other hand, other studies that also leverage BERT and its variants have shown that crowd-sourced datasets may contain examples based on different labeling rationales, making automatic classification attempts even harder [18]. To overcome this limitation and improve classification accuracy, in our previous work we evaluated the performance of approaches based on few-shot learning using a reduced training set composed of issues with manually-verified labels [17]. However, manual annotation remains a time-consuming task, even if done on a small set of manually curated examples. Hence, it is desirable to minimize related efforts as much as possible.

With the advent of recent GPT-like LLMs – which exhibit surprising capabilities, even for tasks they are not specifically trained on – researchers have started investigating to what extent it is possible to leverage their potential in addressing software engineering challenges [23, 34]. In this context, generative LLMs leveraging billions parameters, like GPT3.5-turbo [50], have already demonstrated their ability to generate code, provide code completions, and even assist with bug detection and code refactoring [34].

In pursuit of a better understanding of how GPT-like LLMs can be leveraged in automated issue labeling, we aim to investigate the following research question.

RQ: *To what extent we can leverage GPT-like LLMs to classify issue reports?*

To address it, we first explore a zero-shot learning scenario where GPT-like LLMs are prompted without providing any labeled data as an example. In line with prior work [84], our prompt simply includes the input text (i.e., the text content from the issue to be classified), the task description (i.e., the definition of the classes *bug*, *enhancement*, *documentation*, and *questions* used for the issue labeling task) and the desired output format. Previous studies demonstrated that LLMs exhibit variable behaviors also based on the number of examples provided [48, 84]. As such, we also investigate the performance of few-shot learning, using examples of issues in the prompt. Finally, we compare the performance of classifiers based on GPT-like LLMs with fine-tuned BERT-like LLMs. The replication package for this study is publicly available for reuse [1].

Inspired by previous work on prompt-based labeling in low-resource settings [74], we also assess to what extent we can leverage GPT-like LLMs to reduce the costs associated with human annotation. To this aim, we measure the agreement between the model and human annotators.

The main contributions of this study can be summarized as follows. First, we enhance the current understanding of how GPT-like LLMs can be leveraged to address the task of automatic issue report classification. Then, we provide evidence that generative AI can be used to reduce labeling costs for building reliable gold-standard datasets.

The remainder of this paper is organized as follows. In Section 2, we provide background information on issue report classification, LLMs, and their use in software engineering research. In Section 3, we present the methodology adopted in our study. We outline the study results in Section 4 and discuss our findings in Section 5. Finally, we conclude our work in Section 6.

2 BACKGROUND AND RELATED WORK

2.1 Large Language Models

Thanks to the recent improvements in hardware capability (e.g., the accessibility of GPUs), Large Language Models (LLMs) based on the transformer architecture [70] have introduced a significant advancement in the field of Natural Language Processing (NLP) and currently represent the state of the art for many NLP tasks [21, 51, 85]. Their introduction led to numerous subsequent developments and improvements by leveraging the concept of self-attention, which allows the model to attend to different parts of the input sequence when computing the representation of each token. This innovation has made it possible to capture long-range dependencies between tokens, which was not possible with previous architectures, such as RNNs [60] and LSTMs [33]. The key to the success of LLMs is the pre-training phase, which allows the model to learn the language structure from a large corpus of unlabeled data [21, 57].

Many LLMs have been proposed based on the transformer architecture. Among encoder-only models, BERT [21] is built using two different training objectives: masked language modeling and next-sentence prediction. The first objective consists of masking some tokens in the input sequence and then predicting them based on the context. The second objective consists of predicting whether two sentences follow each other in the original text. After BERT was released, many variants have been proposed [30, 45, 61, 88]. After learning to model a language, BERT-like models can be further fine-tuned on downstream tasks, achieving state-of-the-art performance [21, 88]. The success of BERT-like models has fostered research in this field, thus leading to the development of larger models. It is the case of decoder-only models, like GPT, that include billions of parameters.

There is no consensus on the terminology about LLMs. As an example, Yang et al. [79] distinguish between BERT-style models, which are encoder-only or encoder-decoder models, and GPT-style models, which are decoder-only models. Zhao et al. [85], instead, distinguish between pre-trained language models (PLMs) and LLMs based solely on the size of the model instead of the architecture. In fact, they refer to LLMs also as “large-sized PLMs”. Pan et al. [54] make no distinction between PLMs and LLMs, but they categorize them based on their architecture. They divide them into three categories: encoder-only models, such as BERT and his variants, encoder-decoder models, such as T5 [58] and BART [46] or decoder-only models, such as GPT [57], XLNet [80]. For the remainder of the paper, we will refer to encoder-only models as *BERT-like* models, and to decoder-only models as *GPT-like* models.

GPT-like models, – e.g., the GPT family [7, 50, 51], LLaMA [65, 66], PaLM [14], Bard [28], Vicuna [13], Falcon [3], Mistral [37], Zephyr [67] – have demonstrated the power and versatility of the transformer architecture when scaling up the number of parameters [42]. In particular, they exhibit emergent abilities that arise suddenly at large scales and cannot be extrapolated from smaller models. These include few-shot learning on diverse NLP tasks, multi-step reasoning, question answering using world knowledge, and instruction following. The mechanisms behind emergence are not fully understood, but hypothesized factors include model capacity, depth, and ability to leverage huge amounts of pre-training data [76]. Many of those models, after their pre-training phase, are further trained to follow instructions through Reinforcement Learning for Human Feedback (RLHF) [15, 52, 89], a technique for training models to align with human goals by providing feedback in the form of rewards [15]. It is the case of GPT-4 [51], GPT3.5-turbo [50] and their predecessor InstructGPT [52].

After the release of such models, many researchers have started to explore their potential applications in the field of software engineering (SE). In this context, models like GPT3.5-turbo [50], GPT-4 [51], CodeX [12] and Copilot [26] have already demonstrated a surprising ability to generate code, provide code completions, and even assist with bug detection and code refactoring [8, 12]. These

[1]https://figshare.com/s/57739c103f4f7bc61032

models have a huge impact on software development. GitHub declared that 46% of the developers' code was written by Copilot, leading developers to code faster than before [23, 27]. GPT-4 can resolve programming tasks without further training and without adding related examples in the prompt, with an accuracy of 67% [23, 51].

GPT-like models have been already used in the SE field for several tasks, such as program repair [9, 10, 44, 63], code summarization [83], software testing [72], natural language to code [82], code clone detection [22], and code comprehension [81]. Still, GPT-like models require further investigation as several open issues have been reported in the literature and require further investigation [23].

2.2 Issue Report Classification

Nowadays, issue-tracking systems (ITS) (e.g. GitHub[2], GitLab[3], Jira[4]) are widely adopted in software development projects to manage the evolution of software products. ITS can lower the barrier to entry for new contributors, as they provide a simple and effective way to report bugs, request new features, ask questions about the software product, but also find some tasks to work on. However, especially for open source projects, the use of such tools can complicate the work of developers and maintainers, as they have to deal with a large number of issues submitted every day, with different types and quality [5, 24, 56]. Issue reports are used to submit requests for changes, report bugs, or ask questions about the software product. They typically contain a title and a description, the issue status (e.g., open, assigned, closed), a comment thread, and a label indicating the type of issue. Through the issue tracking system, maintainers can assign issues to the most appropriate team member, prioritize them, and track their resolution.

Effectively labeling issues can be helpful for project management and prioritization, as the labels act as a classification and filtering mechanism [19, 47]. Maintainers can also define custom labels to classify issues according to their type, priority, or other characteristics. Labels can provide quick hints about issues, such as what kind of topic an issue is about, what development task the issue is related to, or what priority the issue has.

Previous studies [4, 32] have shown that developers and maintainers often assign an incorrect issue label to the reports. Furthermore, the labeling mechanism is rarely used by contributors at the time the issue is created [5, 39], thus requiring the effort of manually labeling issue reports [24]. To reduce this effort, several approaches to the automatic categorization of issue reports have been proposed. Antoniol et al. [4] leverage traditional machine learning models to distinguish between bugs and any other type of issue. With the same aim, Zhou et al. [86] leveraged both structured and unstructured free-text data. Herzig et al. [32], instead, developed models to distinguish between six categories: bug, feature request, improvement request, documentation request, and others. Following this direction, Kallis et al. introduced Ticket Tagger [39, 40], which represents the textual content of issue titles and bodies using fastText [6] vectors.

[2]https://github.com
[3]https://about.gitlab.com/
[4]https://www.atlassian.com/it/software/jira

Recent progress in the field of NLP has led to the development of LLMs like BERT [21], which have been successfully applied to several SE tasks. These models have been shown to be effective also for the issue report classification problem [2, 18, 36, 73], thus currently representing the state of the art. However, in our previous empirical studies we observed that the performance of these models is impacted by inconsistent and noisy labels, which are common in crowd-sourced datasets [17, 18], in line with evidence provided by Herzig et al. showing that 33.8% of all issue reports are incorrectly labeled [32]. Vargovich et. al. [69] proposed a recommendation system that suggests issues to the contributors of open-source software, based on the skills required. The tool works by labeling the issues with API domains, instead of using the type of the issue.

To the best of our knowledge, GPT-like models have not been applied yet to the problem of issue classification, in spite of the promising results reported for many SE tasks [23]. In this study, we aim to fill this gap in the literature by exploring to what extent we can leverage GPT-like models in automatic issue labeling in a low-resource setting, i.e. when a gold standard for fine-tuning is not available.

3 METHODOLOGY

3.1 Dataset

In this study, we experiment with a subset of a publicly available dataset of GitHub issues reports extracted from open-source software projects. Each issue receives a *label*, which represents the classification target. Possible class values are (i) *bug*, indicating that the issue contains a bug report, (ii) *feature*, indicating that the issue contains suggestions for improvements or requests for new features, (iii) *question*, assigned to issues containing users' questions about software usage, and (iv) *documentation*, assigned to issues that suggest improvements, updates, or corrections to a software's documentation. The dataset was collected and distributed in the scope of the challenge on issue classification organized within the 2nd Intl. Workshop on Natural Language-based Software Engineering (NLBSE '23) [41].

Specifically, we use a manually verified subset of the challenge benchmark, which we created and distributed in the scope of our previous work [17]. The dataset [5] was obtained by extracting 400 issue reports from the NLBSE'23 challenge dataset using stratified sampling and then manually labeled, with a substantial agreement (a kappa = 0.739 was reported by the authors of the original study). The dataset is split into two subsets: 200 issue reports for training and 200 for testing. The distribution of the labels is shown in Table 1. As a result of the annotation, the dataset also contains an indication of the items that were *discarded*, a label used to identify the cases for which the annotators couldn't agree on the label to assign, e.g., due to the insufficient text available. The discarded issues are not considered in the evaluation of the models.

3.2 Choice of the LLMs

Several GPT-like LLMs have been recently proposed, with a prevalence of studies leveraging GPT3.5-turbo [53] —i.e., the model that powers ChatGPT at the time of this writing. Therefore, we decided

[5]https://nlbse2023.github.io/, Last accessed: November 2023

Table 1: Distribution of labels of the dataset from Colavito et al. [17], which we use for this study.

	Hand labeling			
Label	Train set		Test set	
Bug	47	24%	53	27%
Feature	60	30%	55	28%
Question	44	22%	47	24%
Documentation	33	17%	32	16%
Discarded	16	8%	13	7%
Total	200		200	

to use GPT3.5-turbo [50] for evaluating the performance of LLMs in the issue classification domain. While running our experiments, OpenAI released new versions of ChatGPT, which we also included in our benchmark. At present, the models adopted in this study are available on the OpenAI API and named respectively: *gpt-3.5-turbo-0301*, *gpt-3.5-turbo-0613*, *gpt-3.5-turbo-16k-0613*. Both the first and the second have a context length of 4097 tokens, while the third supports up to 16385 tokens. For all our experiments, we invoked these models using the OpenAI API.

We decided to compare the performance of models with different context lengths as, in the wild, the length of issue reports can vary considerably and, at times, be quite extensive. This is especially problematic for experiments in the few-shot setting (see Section 3.3) in which – beyond the issue text – prompts include examples for each label from the train set, as shown in Figure 1. This could result in exceeding the allowed number of tokens, thus requiring more or less severe cuts depending on the model context length.

On the other hand, we decided to include two different versions of GPT supporting the same context length – i.e., 4097 tokens – to see if, regardless of this parameter, we would observe improved model performance with the new model version. Indeed, previous research on ChatGPT and GPT-4 [11] have shown that these models may exhibit noticeable changes in their behavior and ability on some tasks when new versions are released. Another reason to experiment with models having a lower context length is that LLMs can easily "lose themselves" when the context becomes too long [48] and provide less accurate responses. By experimenting with models enabling different context lengths, we aim to investigate how this factor influences classification performance.

As for the model configuration, we need the temperature parameter to be low. This is necessary to minimize the likelihood of the language model generating incorrect or unpredictable outputs. Indeed, a low temperature makes language models less random and more deterministic in generating text[1]. For this reason, we set the temperature parameter to 0.

SETFIT baseline. In our previous work, we propose and evaluate few-shot learning for issue classification [17] using SETFIT [68], a framework optimized for the fine-tuning of transformer models like Sentence-BERT (SBERT) [59] on small training sets. Sentence transformers are a family of models that use transformer architectures to generate sentence embeddings. This approach is particularly effective for semantic similarity tasks, for which BERT-like models are not optimized [59].

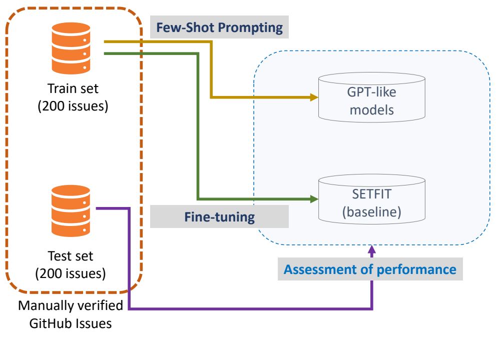


Figure 1: Experimental setting.

The SETFIT model achieved state-of-the-art performance on the manually verified subset of the NLBSE'23 challenge dataset [41] – i.e., the same dataset used in our experiments. For this reason, we choose SETFIT as a baseline for evaluating the performance of GPT-like models in this study. In order to do so, we replicate the SETFIT fine-tuning and evaluation approaches by executing the scripts from the replication package distributed with the original study. [6] In doing so, we use the same train-test partition: specifically, we fine-tune the SETFIT model on the 50% of the gold standard dataset; then, we test the performance of both the SETFIT model and the GPT-like models on the remaining 50% (see Figure 1).

3.3 Prompting GPT-like LLMs

In line with previous work [7, 65, 84], we build prompts for the GPT-like models in order to experiment with ***zero-shot*** and ***few-shot*** settings. In fact, the performance of GPT-like models has been shown to vary significantly based on how the prompt is crafted [84] and on the context length [48]. By comparing the performance achieved in the zero-shot setting with those obtained by adding to the prompt N examples per class (with N either equal to 1 or 2), we aim to assess if the performance of the model improves when an increasing number of examples are provided in combination with the label definition.

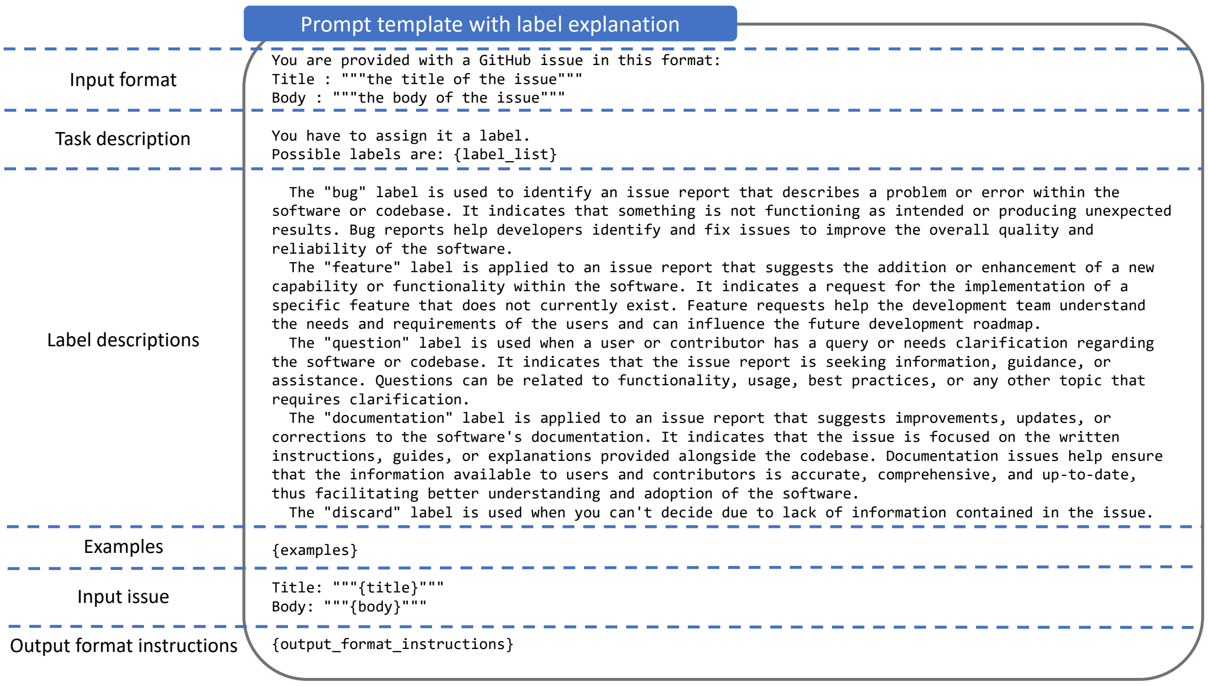
Prompt template. The prompts we used in this study adhere to the template shown in Figure 2, which we describe in the following.

Input format. The format of the issue report provided as input to the model, comprising a *title* and a *body*.

Task Description. The description of the classification task, provided to instruct the model on what to do. It includes the list of possible labels.

Label Descriptions. A list of definitions, one for each class label. To prepare the label descriptions for our issue classification task, we follow a semi-automatic procedure. A first description of the labels is obtained by querying ChatGPT. Then, we manually verify that the descriptions are correct and representative of the four

[6] https://github.com/collab-uniba/Few-Shot-Learning-for-Issue-Report-Classification



```
You are provided with a GitHub issue in this format:
Title : """the title of the issue"""
Body : """the body of the issue"""

You have to assign it a label.
Possible labels are: {label_list}

  The "bug" label is used to identify an issue report that describes a problem or error within the
software or codebase. It indicates that something is not functioning as intended or producing unexpected
results. Bug reports help developers identify and fix issues to improve the overall quality and
reliability of the software.
  The "feature" label is applied to an issue report that suggests the addition or enhancement of a new
capability or functionality within the software. It indicates a request for the implementation of a
specific feature that does not currently exist. Feature requests help the development team understand
the needs and requirements of the users and can influence the future development roadmap.
  The "question" label is used when a user or contributor has a query or needs clarification regarding
the software or codebase. It indicates that the issue report is seeking information, guidance, or
assistance. Questions can be related to functionality, usage, best practices, or any other topic that
requires clarification.
  The "documentation" label is applied to an issue report that suggests improvements, updates, or
corrections to the software's documentation. It indicates that the issue is focused on the written
instructions, guides, or explanations provided alongside the codebase. Documentation issues help ensure
that the information available to users and contributors is accurate, comprehensive, and up-to-date,
thus facilitating better understanding and adoption of the software.
  The "discard" label is used when you can't decide due to lack of information contained in the issue.

{examples}

Title: """{title}"""
Body: """{body}"""

{output_format_instructions}
```

Figure 2: Prompt template. In the zero-shot setting we do not include examples.

issue-report classes considered in this study, i.e., *bug*, *feature*, *documentation*, and *question*. Label descriptions are always provided in the zero-shot experimental setting, to partially compensate for the lack of examples from which a model can learn how to perform a classification. On the other hand, in the few-shot experimental setting, we experiment with both including and omitting label descriptions. In the first case, we keep the definition of the labels and add classification examples. In the second case, we remove the label definitions and only include classification examples. This is done for two reasons: on the one hand, to understand if label explanations are still beneficial when examples are provided to the model; on the other hand, to explore a setting in which more space is reserved for examples within the model context window. In both cases, we experiment with one-shot and two-shot settings —i.e., for each label, we include one or two examples in the prompt, respectively.

Examples. Examples of labeled issues, taken from the train set of our gold standard dataset. The examples are provided to further instruct the model on how to perform the classification task. Noticeably, the *"examples"* part of the prompt is kept empty in the zero-shot setting.

Input Issue. The issue to be classified, provided in the input format declared in the 'Input format' section at the beginning of the prompt.

Output format instructions. The desired format for the model output. Specifically, in the zero-shot setting, we ask the model to output a JSON object containing the predicted label and the reasoning behind its assignment. Indeed, it has been shown that requesting the model to think step-by-step and reason about its answer – a technique known as *Chain-of-Thought prompting* [77] – can boost the performance of LLMs in zero-shot settings [43]. It is important to notice, however, that such reasoning serves as a prompt-engineering strategy and is not used to evaluate the model — i.e., model evaluations are solely based on the labels explicitly predicted by the models. Conversely, in the few-shot learning experimental setting, we only ask the model to provide a label by completing the last line of the prompt, which starts with "`Label:`". In this case, we do not apply the chain-of-thought strategy: to do so, we would need to provide examples of chain-of-thought reasoning for each example issue included in the prompts, which are not available in our dataset.

Dealing with the context length. Input issues as well as the issues provided as examples in the classification prompts may not fit the available context length offered by GPT-like LLMs. To handle this problem, we adopt different strategies for the zero-shot and the few-shot experimental settings.

In the zero-shot setting, when the input issue does not fit the model context length, we cut the exceeding part of the issue body. This choice is in line with the practices adopted in previous studies on issue classification leveraging transformer-based models, such as RoBERTa [18, 35], and SETFIT [17].

In the few-shot setting, the length of the prompt depends not only on the length of input issue but also on the length of the set of examples provided. Thus, in this case, the strategy for dealing with the limited model context length is strictly related to the strategy adopted for the example selection, which can be summarized as follows. First of all, we make sure that examples are always sampled from the training set, i.e., that no examples from the test set are used for building prompts. Then, to select the set of examples to be included in the prompt for the N-shot setting, we go through a random subset of all the possible example combinations containing N issues per class. We stop exploring the combination space when we find the first set of examples that can be fully contained within the available context length together with the rest of the prompt. If no such combination can be found, we use the shortest examples available for each class. Although this may introduce a potential bias toward shorter issue reports, this approach enables us to reserve the maximum number of tokens for the input issue while still providing a set of complete examples. It is important to note that distinctive elements associated with specific issue classes (e.g., a question for the *Question* class) may appear anywhere in the issue body, even at the very end. In case the prompt exceeds the context length despite employing minimal examples, we construct the prompt by truncating the input body as done in the zero-shot setting.

3.4 Evaluation Strategy

We perform 10 experimental runs for each experimental setting and report the average performance of each model evaluated on the test set. Indeed, as in the few-shot setting the examples in the prompt are randomly selected, this ensures a more reliable estimate of the model performance.

Specifically, to address our RQ, we assess and report the performance of the evaluated models in terms of precision, recall, and f1-measure for all the issue classes. This choice is in line with previous work [35, 39] and reflects the standard methodology to assess the performance of text categorization approaches [62]. Precision is the ratio between the true positive and all the predicted items for a given class. Recall represents the ratio of true positives and all items belonging to a given polarity class. F1-measure is computed as the harmonic mean of precision and recall. To assess the overall performance of each classifier, we also compute the micro- and macro-averaged values for precision, recall, and f1-measure, thereby enabling a quick comparison of the overall performance of each model. The overall performance is computed by adopting micro-averaging as an aggregated metric. Furthermore, we report macro-average, i.e., precision and recall are first evaluated locally for each issue class and then globally by averaging the results of the different categories. The reason for providing both values is that micro and macro- averaging may lead to different results in presence of class imbalance. For instance, the performance on classes with few positive training instances is emphasized by macro-averaging. Conversely, micro-averaging tends to be mainly influenced by the performance on the majority class. It is important to note that, whether it is better to optimize by precision, recall, or f1-measure depends on the particular application scenario. As such, we provide a comparison based on the full set of metrics (see Section 4). Additionally, we compute the Cohen's *kappa* [16] and assess the agreement level between the studied GPT-like models and human raters.

4 RESULTS

To answer our RQ, we experimented with different versions of GPT-3.5 in both the zero-shot and few-shot settings. In the following, we report the results of our experiments in the different settings: zero-shot (Table 2), one-shot (Table 3), and two-shot (Table 4). Then, we compare the best-performing model against the SETFIT baseline (see Table 5). In all the tables, we highlight in bold the best f1-measure, both, by class and overall.

Despite providing formatting instructions, the models might struggle to follow them, thus failing to predict a label —i.e. they can produce nonsensical output such as code, a replication of the input text, or they can simply omit the requested label. In case the output contains zero labels or more than one label, we consider these issues as not classified and discard them. To enable a comprehensive assessment of the performance achieved in each setting, we report the percentage of discarded predictions in each results table.

Looking at Table 2, the most recent model with a 16k context length (16k-0613) achieves the overall best performance (f1-micro = .8155, f1-macro = .8095). Also, it is important to note that the 16k model does not output nonsensical predictions (see 'Discarded' in the last row of the table). However, its overall performance is very close to the one observed for the model with a smaller context window (f1-micro = .8133, f1-macro = .8073). Also, the performance reported across the individual classes is comparable, with the higher performance observed for the *Bug* class, while the *Documentation* appears as the more problematic one, due to a lower recall.

As for the few-shot setting, we observe a general tendency to a slight performance decrease as we leverage models with higher context length and a higher number of examples provided in the prompt. Differently from what was observed for the zero-shot setting (see Table 2) the best performance is achieved by the model with the smaller context (4k-0301). Specifically, we observe a f1-micro = .8099 and f1-macro = .8008 in the one-shot classification setting (Table 3) which improves when label explanations are also included in the prompt (see f1-micro = .8158 and f1-macro = .8086 in Table 3). In particular, we observe that keeping the label explanations in the prompt is beneficial for all the settings, with larger improvements for the more recently released models. We also observe that the older model (4k-0301) seems to benefit more from the availability of examples in the prompt compared to the newer models (4k-0613 and 16k-0613). In particular, there is a significant improvement in the metrics for the 4k-0301 model when adding one example for each class in the prompt.

Conversely, the newer models are better at following instructions thus performing better in the zero-shot setting. Conversely, the older model is better at learning from issue data examples in the few-shot setting. Moreover, when adding examples to the prompt and keeping the label explanations, all models tend to output more nonsensical predictions. This is probably due to the fact that the prompt becomes too long and the model struggles in focusing on the relevant information and following the instructions.

As for the comparison with the SETFIT baseline, we include consideration of the 16k-0613 model as this achieves the best performance in terms of a combination of f1-measure and percentage of discarded items due to nonsensical outputs of the model. Specifically, none of the predictions was discarded by this model (see

Table 2). We observe that the zero-shot GPT-3.5 model achieves a slightly lower performance (F1 micro = .8155, F1 macro = .8095) than SETFIT (F1 micro = .8321, F1 macro = .8246), while still being comparable.

As a further assessment of the labeling ability of GPT-like models, we evaluated the agreement between GPT-3.5 and human annotators using Cohen's *kappa*. In Table 6, we report the agreement of the three GPT-like models with the human-provided labels, for all settings. For the interpretation of kappa values, we follow a consolidated interpretation [71] suggesting that the agreement is less than chance if $\leq \kappa \leq 0$, slight if $0.01 \leq \kappa \leq 0.20$, fair if $0.21 \leq \kappa \leq 0.40$, moderate if $0.41 \leq \kappa \leq 0.60$, substantial if $0.61 \leq \kappa \leq 0.80$, and almost perfect if $0.81 \leq \kappa \leq 1$. According to this interpretation, we observe a substantial agreement when using the older model (4k-0301), regardless of the experimental setting. As for the newer models, we observe substantial agreement only when the zero-shot setting is enabled.

5 DISCUSSION

Our results highlight the potential of leveraging GPT-like LLMs for automated issue labeling, while also revealing some limitations. Based on the empirical evidence provided in this paper, we answer our research question by also discussing the implications of our findings for research and practice. Then, we report the threats to validity of this study and how we address them.

Classification performance in different settings (zero-shot vs. few-shot). Classification performance of GPT-like models is known to be affected by the size of the context [48] as well as by the choice of the prompt [7]. As such, in this study, we experimented with models implementing a different context size as well as with different experimental settings for prompt definition, i.e. zero-shot vs. few-shot setting and, within the few-shot setting, by including or not the label definition.

We observe comparable performance in all settings, with no significant differences. In particular, we do not observe an improvement in performance when examples are provided in the prompt (few-shot setting), compared to the setting in which the prompt only contains the definition of labels used for classification (zero-shot setting). Overall, the best performance models achieve f1-micro equal to .8155 (16k-016 in the zero-shot setting), .8158 (4k-0301 in the one-shot learning, when the explanation of labels is provided in the prompt), and .8159 (4k-0301 in the two-shot learning, with label explanations). Furthermore, we observe that keeping the label explanation is useful to yield better performance for the 1-2-shot settings. However, we also observe the models give more non-sense responses in such settings.

In the zero-shot setting, the best performance is achieved by the more recently released, longer context model (16k). Conversely, when examples are provided in the few-shot learning, we observe the best performance with the oldest model, using a context of 4k. However, the difference in performance between the settings is negligible.

GPT-like vs. BERT-like models. Our results demonstrate that GPT-like models are able to achieve a performance comparable to the state of the art represented by BERT-like models. In particular, in this study, we compare the performance of GPT-like models with the baseline [17] that uses SETFIT for few-shot fine-tuning of BERT-like models. For the sake of comparison with the SETFIT baseline, we select the performance of the GPT-like model obtained in the zero-shot learning with the 16k model, as this is the one for which no output was discarded (see Table 5).

The GPT-like model achieves comparable performance (f1 micro = .8155, f1-macro = .8095) to the SETFIT baseline (f1 micro = .8321, f1-macro = .8246). It is important to note that the SETFIT model was fine-tuned on a subset of the issue report gold standard dataset (see Figure 1), while GPT-3.5 was used in a zero-shot setting and without performing fine-tuning. This suggests that GPT-3.5 can be used to classify issue reports without any task-specific fine-tuning with a negligible loss in performance compared to BERT-like models, which is a significant advantage in the absence of labeled data.

While not directly comparable due to the use of different datasets, we also discuss the comparison with previous work leveraging BERT-like LLMs for issue classification. Most of the recent papers were published in the scope of an issue-labeling competition [38]. All classifiers participating in the evaluation campaign implemented supervised approaches by fine-tuning BERT and its variants using the challenge dataset of Github issues collected by the organizers. These issues were annotated using three classes, namely *Bug*, *Enhancement*[7], and *Question*, based on the label provided by either the project contributors or the users. All the proposed classifiers achieved an overall f1-measure higher than .82. We can conclude that, despite we assess them on four classes rather than three, GPT-like models achieve comparable performance to state-of-the-art BERT-like models. However, while the overall classification performance is comparable, the BERT-like LLMs report a lower performance of the minority class *question*, for which they achieve $.447 \leq f1 \leq .692$. This problem is addressed when using SETFIT on a small, manually validated, and equally distributed dataset [17], which achieves f1 = .8528 for *question* (see Table 5). Similarly, the GPT-like models report more consistent performance across the various classes, without requiring a fine-tuning step, thus representing a valid approach whenever a gold standard dataset is not available for training.

Reducing the Cost of Annotation with GPT-like LLMs. Developing and maintaining software modules that leverage LLMs can be problematic, in spite of their huge potential and unquestionable capabilities. In fact, deploying LLMs for classification purposes might not be a feasible option when computational resources are limited. In such cases, building traditional machine-learning classifier models might be a viable solution. However, this requires building a gold standard dataset for training. In our study, we observe a Cohen's kappa > .7 in all settings, thus indicating a substantial agreement between GPT-3.5 and human annotators. Then, in line with recommendations of previous work [31, 74, 87], we suggest using a GPT-like model as a part of an annotation team, to reduce labeling costs associated with human annotation [74] or for data augmentation [20, 49]. Alternatively, it can be used to sample data for human annotation when a high imbalance is expected by applying random sampling to a data source. It is the case, for example, of the imbalance observed in crowdsourced datasets [38, 41] for issue labeling, for which a majority of *bug* is observed. In these

[7] *Enhancement* was renamed as *Feature* in the NLBSE'23 edition of the challenge.

Table 2: Performance in the zero-shot classification setting with GPT-3.5. The best f1 by class and overall is highlighted in bold.

	4k-0301			4k-0613			16k-0613			
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
Bug	0,6684	0,9509	0,7850	0,7100	0,9792	0,8232	0,7133	0,9811	**0,8261**	53
Documentation	0,9488	0,5511	0,6971	0,8627	0,6149	0,7180	0,8853	0,6191	**0,7285**	32
Feature	0,8916	0,8018	0,8442	0,8883	0,8527	**0,8701**	0,8861	0,8491	0,8672	55
Question	0,7849	0,8438	0,8129	0,8869	0,7594	**0,8181**	0,8668	0,7719	0,8164	47
Micro avg	0,7899	0,7882	0,7891	0,8137	0,8128	0,8133	0,8155	0,8155	**0,8155**	187
Macro avg	0,8234	0,7869	0,7848	0,8370	0,8016	0,8073	0,8379	0,8053	**0,8095**	187
Discarded prediction		0,2%			0,11%			–		

Table 3: Performance in the one-shot classification setting with GPT-3.5. The best f1 by class and overall is highlighted in bold.

	4k-0301			4k-0613			16k-0613			
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
(a) Without label explanations in the prompt										
Bug	0,7212	0,9792	**0,8305**	0,7115	0,8302	0,7658	0,7049	0,8358	0,7643	53
Documentation	0,9360	0,6128	**0,7400**	0,8500	0,5745	0,6839	0,8380	0,5553	0,6674	32
Feature	0,8541	0,8491	**0,8513**	0,8862	0,6564	0,7524	0,9105	0,6582	0,7625	55
Question	0,8316	0,7375	**0,7813**	0,5204	0,8438	0,6416	0,5296	0,8688	0,6569	47
Micro avg	0,8123	0,8075	**0,8099**	0,7175	0,7171	0,7173	0,7195	0,7187	0,7191	187
Macro avg	0,8357	0,7947	**0,8008**	0,7420	0,7262	0,7109	0,7458	0,7295	0,7128	187
Discarded prediction		0,59%			0,05%			0,11%		
(b) With label explanations in the prompt										
Bug	0,7332	0,9792	**0,8385**	0,7018	0,8774	0,7791	0,7006	0,8962	0,7863	53
Documentation	0,9208	0,6426	**0,7566**	0,8196	0,6234	0,7074	0,8310	0,6064	0,7010	32
Feature	0,8664	0,8255	**0,8453**	0,8750	0,6709	0,7582	0,8687	0,6909	0,7689	55
Question	0,8570	0,7406	**0,7939**	0,6454	0,7875	0,7066	0,6516	0,7594	0,7006	47
Micro avg	0,8231	0,8086	**0,8158**	0,7502	0,7374	0,7437	0,7546	0,7396	0,7470	187
Macro avg	0,8443	0,7970	**0,8086**	0,7604	0,7398	0,7378	0,7630	0,7382	0,7392	187
Discarded prediction		1,77%			1,71%			1,98%		

Table 4: Results of the two-shot classification setting with GPT-3.5. The best f1 by class and overall is highlighted in bold.

	4k-0301			4k-0613			16k-0613			
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
(a) Without label explanations in the prompt										
Bug	0,7394	0,9623	**0,8361**	0,7151	0,8132	0,7606	0,7187	0,8132	0,7627	53
Documentation	0,9335	0,6532	**0,7683**	0,7383	0,6319	0,6803	0,7279	0,6085	0,6618	32
Feature	0,8640	0,8182	**0,8402**	0,9152	0,4764	0,6254	0,9112	0,5055	0,6490	55
Question	0,7972	0,7906	**0,7930**	0,4905	0,8750	0,6278	0,4847	0,8625	0,6201	47
Micro avg	0,8181	0,8128	**0,8154**	0,6806	0,6791	0,6799	0,6797	0,6797	0,6797	187
Macro avg	0,8335	0,8061	**0,8094**	0,7148	0,6991	0,6735	0,7106	0,6974	0,6734	187
Discarded prediction		0,64%			0,21%			–		
(b) With label explanations in the prompt										
Bug	0,7351	0,9604	**0,8325**	0,7199	0,8472	0,7779	0,7112	0,8585	0,7774	53
Documentation	0,9327	0,6617	**0,7731**	0,6763	0,6553	0,6643	0,6481	0,6468	0,6469	32
Feature	0,8723	0,8145	**0,8419**	0,8893	0,5691	0,6934	0,8985	0,5600	0,6893	55
Question	0,8068	0,7813	**0,7934**	0,6113	0,7875	0,6873	0,6214	0,7688	0,6854	47
Micro avg	0,8201	0,8118	**0,8159**	0,7160	0,7070	0,7114	0,7089	0,7021	0,7055	187
Macro avg	0,8367	0,8045	**0,8102**	0,7242	0,7148	0,7057	0,7198	0,7085	0,6998	187
Discarded prediction		1,02%			0,86%			0,96%		

Table 5: Comparison between SETFIT and GPT-3.5.

	SETFIT			GPT-3.5 (16k-0613), zero-shot			
Label	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
Bug	0.8723	0.8472	**0.8590**	0,7133	0,9811	0,8261	53
Documentation	0.9039	0.6594	**0.7616**	0,8853	0,6191	0,7285	32
Feature	0.7494	0.9182	0.8251	0,8861	0,8491	**0,8672**	55
Question	0.8754	0.8319	**0.8528**	0,8668	0,7719	0,8164	47
Micro average	0.8321	0.8321	**0.8321**	0,8155	0,8155	0,8155	187
Macro average	0.8502	0.8142	**0.8246**	0,8379	0,8053	0,8095	187

Table 6: Agreement between GTP-like models and gold labels provided by human annotators. We highlight in bold the best values for Cohen's *kappa*.

		Cohen's *kappa*		
Experimental setting	Label Explanation	4k-0301	4k-0613	16k-0613
0-Shot	Y	**0,7045**	0,6806	0,6838
1-Shot	Y	**0,7102**	0,5705	0,5590
1-Shot	N	**0,7163**	0,5846	0,5839
2-Shot	Y	**0,7247**	0,4984	0,5025
2-Shot	N	**0,7157**	0,5515	0,5605

cases, the annotation set for manual labeling can be extracted using GTP-like models for labeling crowdsourced data using opportunistic sampling, to mitigate the problem of class imbalance. In fact, a desirable property of a training set is that its items are equally distributed across the existing classes of values [29]. By applying opportunistic sampling to the creation of an annotation set based on labels provided by GPT-like models, the risk of obtaining an unbalanced dataset could be reduced. Nonetheless, this could not be always feasible, due to the licensing of the GPT-family models. These limitations could be addressed by using open-source models, such as LLaMa2[66], Vicuna [13], Falcon [3], Mistral [37], and Zephyr [67], which we plan to investigate in our future studies.

5.1 Threats to Validity

One potential threat to the *internal validity* of our study could reside in the choice and design of the prompt templates. Indeed, previous work demonstrated how prompt engineering is crucial to optimize the classification performance of GTP-like models [7]. To mitigate this threat we experiment with different prompts of various lengths, i.e. by including the label definition only (zero-shot), or by also appending examples of labeled issues (few-shot). Additionally, we examined the influence of the number of shots in the few-shot setting.

Another potential limitation is associated with the impossibility of verifying that the Github issues we used in our study have not been included in the training data for GPT-like models by their authors, as these are distributed for use as black boxes. Future replications will have to involve consideration of open-source models for which the training dataset is accessible (e.g., LLaMa2[66], Falcon [3], Mistral [37], Zephyr [67]), to properly address this concern and control for data leakage.

Furthermore, another threat to internal validity concerns internal factors such as the configuration of the parameters, which is a known limitation of any ML-based approach. Specifically, when using GPT-like models, the model temperature has to be set, as high values in temperature might result in more unpredictable output [1]. While this can be seen as an advantage in creative tasks (e.g., supporting dialogue-based interaction), it could be detrimental for classification tasks as the model can produce highly divergent outputs when the same prompt is provided [78]. In this study, we control for this factor by setting a unique temperature value for all the experimental conditions. Future studies might consider investigating the impact of different values of temperature for classification tasks.

Conclusion validity is influenced by the choice of datasets to include in our benchmark. In our study, we selected a subset of the dataset mined by the software engineering community for the NLBSE23 Tool Competition [41]. We manually validated the labels in this subset in the scope of our previous work [17], thus ensuring that the labels are of high quality. However, we acknowledge that our methodology could produce different results if applied to different datasets. Furthermore, our findings may not necessarily generalize to data from other platforms, different from Github. In this perspective, the choice of focusing on Github issues might have limited the *external validity* of our findings, thus calling for further replications to confirm our conclusions.

Finally, as a further limitation to the external validity, we acknowledge that the choice of focusing on GPT-like models released by OpenAI further limits the generalizability of our findings. We advocate in favor of future replications, also including consideration of other – possibly open-source – models, such as LLaMa2[66], Falcon [3], Mistral [37], and Zephyr [67].

6 CONCLUSION

In this study, we explore the performance of GPT-like models for automatic issue classification. Specifically, we investigated to what extent we can leverage GPT-like LLMs to achieve state-of-the-art performance while reducing the costs associated with the human annotation of issues. Our study is exploratory in nature and aims at investigating the opportunities of leveraging generative LLMs in low-resource settings, i.e. when a gold standard is not available for fine-tuning state-of-the-art approaches based on BERT-like models. It is the case, for example, of (i) new projects for which issues are not available yet for annotation, or (ii) existing projects for which resources are not available to support the time-consuming task of manual creation of a gold standard dataset.

Our empirical results show that GPT-like LLMs can achieve a performance comparable to the state-of-the-art BERT-like LLMs, without the need for fine-tuning. As such, whenever a gold standard dataset is not available, the issue classification task can still be addressed successfully. Since we do not observe a performance improvement when examples are provided in the prompt (few-shot setting), we conclude that a zero-shot approach is sufficient to achieve satisfying performance. Furthermore, the observed substantial agreement with human raters suggests that GPT-like models can be also used to complement costly human annotation when creating a gold standard dataset for supervised learning, whether based on traditional machine learning or the fine-tuning of BERT-like models. However, license limitations can represent an issue for proprietary GPT-like LLMs.

As a final consideration, we highlight that using LLMs for building classifiers might pose important deployment challenges due to computational and scalability issues. A cost analysis, in terms of computational resources, should always be conducted to assess whether deploying LLMs is a feasible solution in practice. In our setting, we were not required to deploy the GPT-like models locally, as they were invoked through API calls. However, this setting might not be representative of several other application scenarios, as relying on managed LLM instances may introduce unwanted external dependencies and trigger privacy-related concerns —e.g., in case of issues containing sensitive data. In such cases, on-premise deployment might be required. In future work, we plan to replicate the current study by also assessing the capabilities of open-source LLMs for issue classification.

Future replications with larger and more varied benchmark datasets – including issues from different projects and platforms – are required as well. These will enable evaluating the generalizability of our findings, thus assessing how consistent is the behavior of zero-shot classification across different models and datasets. Ultimately, conducting such an extended comparison with other LLMs and datasets would further our understanding of the potential limitations of this approach, providing valuable insights for future research on this topic.

ACKNOWLEDGMENTS

This research was co-funded by the NRRP Initiative, Mission 4, Component 2, Investment 1.3 - Partnerships extended to universities, research centres, companies and research D.D. MUR n. 341, 15.03.2022 – Next Generation EU ("FAIR - Future Artificial Intelligence Research", code PE00000013, CUP H97G22000210007), the Complementary National Plan PNC-I.1 - Research initiatives for innovative technologies and pathways in the health and welfare sector - D.D. 931 of 06/06/2022 ("DARE - DigitAl lifelong pRevEntion initiative", code PNC0000002, CUP B53C22006420001), and by the PRIN 2022 call of MUR ("QualAI: Continuous Quality Improvement of AI-based Systems", CUP H53D23003510006).

REFERENCES

[1] [n. d.]. OpenAI API Documentation, How should I set the temperature parameter? https://platform.openai.com/docs/guides/text-generation/how-should-i-set-the-temperature-parameter. Accessed: 2023-11-17.

[2] W. Alhindi, A. Aleid, I. Jenhani, and M. Mkaouer. 2023. Issue-Labeler: an ALBERT-based Jira Plugin for Issue Classification. In *2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE Computer Society, Los Alamitos, CA, USA, 40–43. https://doi.org/10.1109/MOBILSoft59058.2023.00012

[3] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance. (2023).

[4] Giuliano Antoniol, Kamel Ayari, Massimiliano Di Penta, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2008. Is it a bug or an enhancement? a text-based approach to classify change requests. In *Proceedings of the 2008 Conf. of the center for advanced studies on collaborative research: meeting of minds*. ACM, New York, NY, USA. https://doi.org/10.1145/1463788.1463819

[5] Tegawendé F. Bissyandé, David Lo, Lingxiao Jiang, Laurent Réveillère, Jacques Klein, and Yves Le Traon. 2013. Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub. In *2013 IEEE 24th Int'l Symposium on Software Reliability Engineering (ISSRE)*. https://doi.org/10.1109/ISSRE.2013.6698918

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models Are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.

[8] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712 [cs.CL]

[9] Jialun Cao, Meiziniu Li, Ming Wen, and Shing chi Cheung. 2023. A study on Prompt Design, Advantages and Limitations of ChatGPT for Deep Learning Program Repair. arXiv:2304.08191 [cs.SE]

[10] Yiannis Charalambous, Norbert Tihanyi, Ridhi Jain, Youcheng Sun, Mohamed Amine Ferrag, and Lucas C. Cordeiro. 2023. A New Era in Software Security: Towards Self-Healing Software via Large Language Models and Formal Verification. arXiv:2305.14752 [cs.SE]

[11] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is ChatGPT's behavior changing over time? arXiv:2307.09009 [cs.CL]

[12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]

[13] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[14] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. arXiv:2204.02311 [cs.CL]

[15] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances*

in neural information processing systems 30 (2017).
[16] Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin* 70, 4 (1968), 213. https://doi.org/10.1037/h0026256
[17] Giuseppe Colavito, Filippo Lanubile, and Nicole Novielli. 2023. Few-Shot Learning for Issue Report Classification. In *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*. 16–19. https://doi.org/10.1109/NLBSE59153.2023.00011
[18] Giuseppe Colavito, Filippo Lanubile, and Nicole Novielli. 2023. Issue Report Classification Using Pre-Trained Language Models. In *Proceedings of the 1st International Workshop on Natural Language-Based Software Engineering* (Pittsburgh, Pennsylvania) *(NLBSE '22)*. Association for Computing Machinery, New York, NY, USA, 29–32. https://doi.org/10.1145/3528588.3528659
[19] Javier Luis Cánovas Izquierdo, Valerio Cosentino, Belén Rolandi, Alexandre Bergel, and Jordi Cabot. 2015. GiLA: GitHub label analyzer. In *2015 IEEE 22nd Int'l Conf.on Software Analysis, Evolution, and Reengineering (SANER)*. https://doi.org/10.1109/SANER.2015.7081860
[20] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. AugGPT: Leveraging ChatGPT for Text Data Augmentation. arXiv:2302.13007 [cs.CL]
[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423
[22] Shihan Dou, Junjie Shan, Haoxiang Jia, Wenhao Deng, Zhiheng Xi, Wei He, Yueming Wu, Tao Gui, Yang Liu, and Xuanjing Huang. 2023. Towards Understanding the Capability of Large Language Models on Code Clone Detection: A Survey. arXiv:2308.01191 [cs.SE]
[23] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large Language Models for Software Engineering: Survey and Open Problems. arXiv:2310.03533 [cs.SE]
[24] Qiang Fan, Yue Yu, Gang Yin, Tao Wang, and Huaimin Wang. 2017. Where Is the Road for Issue Reports Classification Based on Text Mining?. In *2017 ACM/IEEE Int'l Symposium on Empirical Software Engineering and Measurement (ESEM)*. https://doi.org/10.1109/ESEM.2017.19
[25] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1536–1547. https://doi.org/10.18653/v1/2020.findings-emnlp.139
[26] GitHub. 2021. GitHub Copilot. https://github.com/features/copilot
[27] GitHub. 2023. GitHub Copilot now has a better AI model and new capabilities. https://github.blog/2023-02-14-github-copilot-now-has-a-better-ai-model-and-new-capabilities/
[28] Google. 2023. Bard. https://bard.google.com/chat
[29] Haibo He and Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284. https://doi.org/10.1109/TKDE.2008.239
[30] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. arXiv:2006.03654 [cs.CL]
[31] Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. AnnoLLM: Making Large Language Models to Be Better Crowdsourced Annotators. arXiv:2303.16854 [cs.CL]
[32] Kim Herzig, Sascha Just, and Andreas Zeller. 2013. It's not a bug, it's a feature: How misclassification impacts bug prediction. In *2013 35th International Conference on Software Engineering (ICSE)*. 392–401. https://doi.org/10.1109/ICSE.2013.6606585
[33] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (11 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
[34] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2023. Large Language Models for Software Engineering: A Systematic Literature Review. arXiv:2308.10620 [cs.SE]
[35] Maliheh Izadi. 2022. CatIss: An Intelligent Tool for Categorizing Issues Reports using Transformers. In *(NLBSE 2022)*. https://doi.org/10.1145/3528588.3528662
[36] Maliheh Izadi, Kiana Akbari, and Abbas Heydarnoori. 2022. Predicting the objective and priority of issue reports in software repositories. *Empirical Software Engineering* 2 (2022). https://doi.org/10.1007/s10664-021-10085-3
[37] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]
[38] Rafael Kallis, Oscar Chaparro, Andrea Di Sorbo, and Sebastiano Panichella. 2022. NLBSE'22 Tool Competition. In *Proceedings of The 1st Intl. Workshop on Natural Language-based Software Engineering (NLBSE'22)*.
[39] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2019. Ticket Tagger: Machine Learning Driven Issue Classification. In *2019 IEEE Intl. Conf. on Software Maintenance and Evolution, ICSME 2019, Cleveland, OH, USA, September 29 - October 4, 2019*. IEEE. https://doi.org/10.1109/ICSME.2019.00070
[40] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2021. Predicting issue types on GitHub. *Science of Computer Programming* (2021). https://doi.org/10.1016/j.scico.2020.102598
[41] Rafael Kallis, Maliheh Izadi, Luca Pascarella, Oscar Chaparro, and Pooja Rani. 2023. The NLBSE'23 Tool Competition. In *Proceedings of The 2nd Intl. Workshop on Natural Language-based Software Engineering (NLBSE'23)*.
[42] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG]
[43] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 22199–22213. https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf
[44] Márk Lajkó, Viktor Csuvik, and László Vidács. 2022. Towards JavaScript program repair with Generative Pre-trained Transformer (GPT-2). In *2022 IEEE/ACM International Workshop on Automated Program Repair (APR)*. 61–68. https://doi.org/10.1145/3524459.3527350
[45] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:1909.11942 [cs.CL]
[46] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703
[47] Zhifang Liao, Dayu He, Zhijie Chen, Xiaoping Fan, Yan Zhang, and Shengzong Liu. 2018. Exploring the characteristics of issue-related behaviors in GitHub using visualization techniques. *IEEE Access* (2018). https://doi.org/10.1109/ACCESS.2018.2810295
[48] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172 [cs.CL]
[49] Anders Giovanni Møller, Jacob Aarup Dalsgaard, Arianna Pera, and Luca Maria Aiello. 2023. Is a prompt and a few samples all you need? Using GPT-4 for data augmentation in low-resource classification tasks. arXiv:2304.13861 [cs.CL]
[50] OpenAI. 2022. ChatGPT: Optimizing Language Models for Dialogue.
[51] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]
[52] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.
[53] Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. arXiv:2308.02828 [cs.SE]
[54] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying Large Language Models and Knowledge Graphs: A Roadmap. arXiv:2306.08302 [cs.CL]
[55] Nitish Pandey, Debarshi Sanyal, Abir Hudait, and Amitava Sen. 2017. Automated classification of software issue reports using machine learning techniques: an empirical study. *Innovations in Systems and Software Engineering* (12 2017). https://doi.org/10.1007/s11334-017-0294-1
[56] Sebastiano Panichella, Gabriele Bavota, Massimiliano Di Penta, Gerardo Canfora, and Giuliano Antoniol. 2014. How Developers' Collaborations Identified from Different Sources Tell Us about Code Changes. In *2014 IEEE International Conference on Software Maintenance and Evolution*. https://doi.org/10.1109/ICSME.2014.47
[57] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245, LastaccessedNov.2023
[58] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21, 1, Article 140 (jan 2020), 67 pages.
[59] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang,

Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[60] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.

[61] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL]

[62] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *Comput. Surveys* (2002). https://doi.org/10.1145/505282.505283

[63] D. Sobania, M. Briesch, C. Hanna, and J. Petke. 2023. An Analysis of the Automatic Bug Fixing Performance of ChatGPT. In *2023 IEEE/ACM International Workshop on Automated Program Repair (APR)*. IEEE Computer Society, Los Alamitos, CA, USA, 23–30. https://doi.org/10.1109/APR59189.2023.00012

[64] Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020. Code and Named Entity Recognition in StackOverflow. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. https://www.aclweb.org/anthology/2020.acl-main.443/

[65] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[66] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[67] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct Distillation of LM Alignment. arXiv:2310.16944 [cs.LG]

[68] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient Few-Shot Learning Without Prompts. https://doi.org/10.48550/arXiv.2209.11055

[69] Joseph Vargovich, Fabio Santos, Jacob Penney, Marco A. Gerosa, and Igor Steinmacher. 2023. GiveMeLabeledIssues: An Open Source Issue Recommendation System. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 402–406. https://doi.org/10.1109/MSR59073.2023.00061

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[71] Anthony Viera and Joanne Garrett. 2005. Understanding Interobserver Agreement: The Kappa Statistic. *Family medicine* 37 (06 2005), 360–3.

[72] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2023. Software Testing with Large Language Model: Survey, Landscape, and Vision. arXiv:2307.07221 [cs.SE]

[73] Jun Wang, Xiaofang Zhang, and Lin Chen. 2021. How well do pre-trained contextual language representations recommend labels for GitHub issues? *Knowledge-Based Systems* (2021). https://doi.org/10.1016/j.knosys.2021.107476

[74] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want To Reduce Labeling Cost? GPT-3 Can Help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 4195–4205. https://doi.org/10.18653/v1/2021.findings-emnlp.354

[75] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. 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*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 8696–8708. https://doi.org/10.18653/v1/2021.emnlp-main.685

[76] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs.CL]

[77] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 24824–24837. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[78] Frank F. Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A Systematic Evaluation of Large Language Models of Code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming* (San Diego, CA, USA) *(MAPS 2022)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3520312.3534862

[79] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. arXiv:2304.13712 [cs.CL]

[80] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[81] Zhiqiang Yuan, Junwei Liu, Qiancheng Zi, Mingwei Liu, Xin Peng, and Yiling Lou. 2023. Evaluating Instruction-Tuned Large Language Models on Code Comprehension and Generation. arXiv:2308.01240 [cs.CL]

[82] Daoguang Zan, Bei Chen, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Wang Yongji, and Jian-Guang Lou. 2023. Large Language Models Meet NL2Code: A Survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 7443–7464. https://doi.org/10.18653/v1/2023.acl-long.411

[83] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2020. Retrieval-Based Neural Source Code Summarization. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, New York, NY, USA, 1385–1397. https://doi.org/10.1145/3377811.3380383

[84] Ting Zhang, Ivana Clairine Irsan, Ferdian Thung, and David Lo. 2023. Revisiting Sentiment Analysis for Software Engineering in the Era of Large Language Models. arXiv:2310.11113 [cs.SE]

[85] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL]

[86] Yu Zhou, Yanxiang Tong, Ruihang Gu, and Harald C. Gall. 2014. Combining Text Mining and Data Mining for Bug Report Classification. *2014 IEEE International Conference on Software Maintenance and Evolution* (2014).

[87] Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. 2023. Can ChatGPT Reproduce Human-Generated Labels? A Study of Social Computing Tasks. arXiv:2304.10145 [cs.AI]

[88] Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A Robustly Optimized BERT Pre-training Approach with Post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, Sheng Li, Maosong Sun, Yang Liu, Hua Wu, Kang Liu, Wanxiang Che, Shizhu He, and Gaoqi Rao (Eds.). Chinese Information Processing Society of China, Huhhot, China, 1218–1227. https://aclanthology.org/2021.ccl-1.108

[89] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-Tuning Language Models from Human Preferences. arXiv:1909.08593 [cs.CL]