On the Validity of Pre-Trained Transformers for Natural Language Processing in the Software Engineering Domain

Julian von der Mosel, Alexander Trautsch, and Steffen Herbold

Abstract—Transformers are the current state-of-the-art of natural language processing in many domains and are using traction within software engineering research as well. Such models are pre-trained on large amounts of data, usually from the general domain. However, we only have a limited understanding regarding the validity of transformers within the software engineering domain, i.e., how good such models are at understanding words and sentences within a software engineering context and how this improves the state-of-the-art. Within this article, we shed light on this complex, but crucial issue. We compare BERT transformer models trained with software engineering data with transformers based on general domain data in multiple dimensions: their vocabulary, their ability to understand which words are missing, and their performance in classification tasks. Our results show that for tasks that require understanding of the software engineering context, pre-training with software engineering data is valuable, while general domain models are sufficient for general language understanding, also within the software engineering domain.

Index Terms—Natural language processing, software engineering, transformers

1 INTRODUCTION

The introduction of the transformer model [1] has permanently changed the field of Natural Language Processing (NLP) and paved the way for modern language representation models such as BERT [2], XLNet [3], and GPT-2 [4]. These models have in common that they use transfer learning in the form of pre-training to learn a general representation of language, which can then be fine-tuned to various downstream tasks. Pre-training is expensive and requires large text corpora. Therefore, most of the available models are pre-trained by large companies on vast amounts of general domain data such as the entire English Wikipedia or the Common Crawl News dataset [5]. While these models achieve remarkable results in a variety of NLP tasks, the learned word representations (embeddings) still reflect the general domain. This is a problem because the meaning of words varies based on context and is therefore domain dependent. Hence, there is a considerable interest in adapting language representation models to different domains. For example, SciBERT [6], BioBERT [7] and ClinicalBERT [8] are BERT models adopted to the bio-medical domain.

In Software Engineering (SE) there are many technical terms that do not exist in other domains and words that have a different meaning within the domain. For example, the word “bug” in the general domain refers to an insect, but within the SE domain it refers to a defect. Similarly, “ant” is an insect in the general domain, but a build tool within SE. Words such as “bug” and “ant” are called polyseme, i.e., words that have different meanings depending on their context. The notion that pre-trained natural language models should be adopted for the SE domain is not new and was already considered by other, e.g., for word2vec embeddings [9] and Named Entity Recognition (NER) with BERT models [10].

A gap within the previous research is the larger context. The work by Efstathiou et al. [9] only established a difference for the by now dated word2vec approach and not for transformer models. Moreover, the focus is solely on the position of a small set of terms within a vector space. This ignores the ability of transformer models to account for the context, which may be able to infer the correct meaning of polysemes. And while Tabassum et al. [10] successfully demonstrated that pre-training with SE data is also useful with transformer models, they used a small variant of the BERT model and did not explore why the model yielded better results. Thus, it is unclear how much better such a model is at capturing the correct meaning of words and if larger transformer models, as they are usually used within the state-of-the-art of NLP, would perform.

Within this work, we want to close this gap through an exploratory study that aims to answer the following research questions.

RQ1: Are large pre-trained NLP models for SE able to capture the meaning of SE vocabulary correctly?
RQ2: Do large pre-trained NLP models for SE outperform models pre-trained on general domain data and smaller NLP models that do not require pre-training in SE applications?

We study these research questions using seBERT, a BERT-T_{LARGE} model we pre-trained on textual data from Stack Overflow, GitHub, and Jira Issues and BERToverflow, a BERT_{BASE} model that was pre-trained by Tabassum et al. [10] on data from Stack Overflow.

For the first research question, we evaluate the capability of the models to correctly determine the meaning of terms in three ways: the comparison of the vocabularies of the models; their capability to infer missing words in a list of curated sentences; and the capabilities to infer missing polysymes both in a SE domain corpus and a general domain corpus. Inferring missing words is a task typically referred to as Masked Language Modeling (MLM). Through this, we complement the study by Efstathiou et al. [9] to understand the ability of NLP models for SE to not only create valid embeddings, but even predict the correct words given the context. Such a prediction goes beyond similarity of single words and would be a strong indicator that NLP models for the SE domain should outperform general domain models because they are demonstrably better at capturing the correct meaning.

For the second research question, we study the capability of these models to improve the performance of prediction tasks. We use several classification tasks for this purpose. 1) The prediction of bug issues, similar to Herbold et al. [11]. This task allows us to study how well the models perform with relatively long text for a task that is, in fact, often performed inaccurately by humans [12]. 2) The identification of quality improving commits [13], i.e., a task with very short texts. 3) Sentiment mining, a task where it is was already shown that general domain transformers perform well [14].

Through our study, we provide the following contributions to the literature:

- An analysis of the validity of NLP models for SE through a combination of comparison of vocabularies and the prediction of missing words. We show that BERT models trained with SE data are able to capture the correct meaning of SE terminology, at the cost of the correct modeling of some general domain concepts like geographical locations. We further show that larger models trained with more diverse data are better at capturing nuanced meanings than smaller domain specific models.

- We advance the state of issue type prediction and commit intent prediction by improving the model performance significantly with the fine-tuned SE specific BERToverflow and seBERT models. Our data indicates that the improvement is only possibly because the model was pre-trained on SE data and also that larger NLP models outperform smaller models in the SE domain similar to the differences that can be observed between general domain models.

- We found that sentiment mining was not strongly affected by domain specific pre-training and that SE domain transformers perform similar to general domain transformers for this task.

- A basic ethical evaluation of the models revealed that the SE domain models should only be used with care, because they may have some troubling properties, especially with respect to gender bias. This warrants further research and until these aspects are better understood, such models should not be used for any tasks that may be negatively affected by gender bias, though this also warrants caution regarding other biases such as racial bias.

The remainder of this article is structured as follows. We provide some background on transformers in Section 2, followed by a discussion of related work on domain specific NLP models in Section 3. Then, we outline the creation of the SE domain model seBERT in Section 4. Based on these foundations, we introduce our method for validating transformers within the SE domain in Section 5, present the results of this validation in Section 6, and discuss these results in Section 7. Finally, we conclude in Section 8.

2 BACKGROUND ON TRANSFORMERS

NLP is a broad field composed of various disciplines such as computational linguistics, machine learning, artificial intelligence, computer science, and speech processing [15]. Modern NLP models have a variety of applications, including text classification, NER, machine translation, sentiment analysis, and natural language generation. In recent years, transformer models, e.g., BERT [2], BART [16], ALBERT [17], RoBERTa [18], GPT-3 [19], and Switch-C [20] established themselves at the state-of-the-art of NLP within the general domain. Since the focus of our article is on the validity of transformer models within the SE domain, we avoid an in-depth technical discussion of the underlying neural network architectures of these models and refer for this to the literature instead (e.g., [21]). Instead, we outline the concept of such models on a high level in natural language without mathematical details.

Transformers can be described in a single sentence: they are sequence-to-sequence networks with self-attention. Now we only need to understand what the highlighted aspects mean. A sequence-to-sequence network takes as input a sequence and generates a new sequence of the same length. For NLP, the input sequence is usually a tokenized text, possibly augmented with additional tokens, e.g., to represent classes. The input sequence is then encoded, such that each input token is represented by a numeric vector, which means that we have a sequence of vectors through these embeddings. This transformation relies on the self-attention, which means that the tokens are not considered in isolation, but together. Consider this paragraph: the self-attention would consider this whole paragraph at once and, for each word, model how the contextual meaning depends on all other words in the paragraph. This is shown in Fig. 1. The figure shows, e.g., that the meaning of “bug” is strongly influenced by the words “not” and “feature,” and that the meaning of “s” is defined by the apostrophe. Thus, there is no fixed meaning of each word and the meaning depends on the context: “bug” actually means “not bug” or “feature,” “s” is only a contraction of “is”.

That the whole sequence is used as context for each word to understand the contextual meaning is the major difference to previously used NLP models. For example, word
embeddings organize words such that words with similar meanings are close to each other [23]. However, while the context is taken into account when the embedding is calculated, each word only gets a single position. If words are polysemes, the meaning depends on the context, which cannot be accurately encoded with such a fixed embedding. Either one contextual meaning is lost, e.g., “bug” is only close to other insects, but not to words like “defect”. Or different concepts become similar to each other, e.g., “bug” is close to both insects and defects, which means that insects in general are now similar to defects. Moreover, since the embedding does not take the position of a word in an actual sentence into account, the interpretation of sentences based on the grammatical relationship between words is also not possible. Recurrent Neural Networks (RNN) without self-attention, such as Long-Short-Term-Memory (LSTM) networks, can also not take the complete context accurately into account. While such models can also be sequence-to-sequence networks, the relationship between a word at a specific position, and all other words, is linear. Simplified, this means that the influence on the context depends mostly on the distance to a specific word. In comparison, the self-attention learns how each word influences the context, depending on the position without a fixed influence of the distance. For our example from Fig. 1, this means that the long distance between the first “it’s” and the last word “feature” does not mean that there cannot be a strong direct relationship.

Due to these differences, transformers quickly became the most powerful architecture for neural network based NLP. Models based on transformers are the current state of the art. The revolution was spear-headed by BERT [2], followed by similar models such as RoBERTa [18], GPT-3 [19] that were pre-trained with more data, used larger neural networks, and/or optimized the efficiency of the self-attention mechanisms. Such models already gained traction within the SE domain: Zhang et al. [14], who recently established transformers as state-of-the-art for sentiment mining within the SE domain, clearly outperforming smaller sentiment mining models like SentiStrength [24], including models tailored specifically to the SE domain like SentiStrength-SE [25], SentiCR [26], and Senti4SD [27]. Biswas et al. [28] also found a strong performance of BERT for sentiment mining. The drawback of transformer models is the size of the neural networks. Even a relatively small model like BERT-base [2] already has 110 million parameters. The largest currently discussed transformer architectures like GPT-3 [19] and Switch-C [20] have more than 100 billion parameters. Thus, the training of such models requires dedicated hardware and huge amounts of data. To reduce the burden, these models are pre-trained in a self-supervised setting. This means that the neural networks are trained on a large corpus of NLP data, such that the structure of text and meaning of words is known by the network. Self-supervised means that this understanding of the language is trained in a supervised way (i.e., with labels), but that the labels are generated directly from the training data. We will show how this works for the BERT pre-training in Section 4.3. Researchers and practitioners who then want to apply NLP to solve a problem, use these pre-trained models and fine-tune the models for a specific task on a labeled data set. This works with less data and requires only few training steps and is, therefore, usually not computationally expensive. However, this may still require dedicated hardware, due to the size of the models.

3 RELATED WORK ON DOMAIN-SPECIFIC NLP

In recent years there has been considerable interest in adapting pre-trained language models such as word2vec [23], ELMo [29] and BERT to different domains. However, most of the related work concerns the biomedical or financial domain. Pyysalo et al. [30] were among the first to provide domain-specific word2vec embeddings based on biomedical corpora. Since then, recent biomedical adaptations of word2vec and BERT include Dis2Vec [31], BioBERT [7], ClinicalBERT [8] and to some extent SciBERT [6], which has been pre-trained on biomedical and computer science papers. Models such as BioBERT and SciBERT have been shown to outperform BERT in biomedical tasks and achieve state-of-the-art results [6], [7]. The same is true for FinBERT [32] which achieved state-of-the-art results in financial sentiment analysis. However, except for SciBERT, the domain-specific BERT models are not pre-trained from scratch, but use the same vocabulary as BERT or are further pre-trained using BERTs weights.

Efstathiou et al. [9] pre-trained a general-purpose word representation model for the software engineering domain. They trained the so_word2vec on 15 GB of textual data from Stack Overflow posts. The authors compare the so_word2vec model with the original Google news word2vec model and show that it performs well in capturing SE specific meanings and analogies. The main difference with our work is that we consider contextual embeddings and provide a more extensive evaluation of the validity of the resulting embeddings. In addition, our seBERT model was trained using 119 GB of data from multiple sources, i.e., we
also use GitHub issues and commit messages in addition to data from Stack Overflow. Ferrari and Esuli [33] used word embeddings to identify terms that are shared between domains, but which may have different meanings. While their work was not on the difference between the SE domain and the general domain, they showed that word embeddings trained on corpora from different domains indeed yield different embeddings for polysemous word, which provides another indication that general domain models may have problems when used for domain-specific purposes.

Recently, Tabassum et al. [10] proposed a similar approach to ours, in which GloVe, ELMo, and BERT models are pre-trained on 152 M sentences from Stack Overflow. In addition, the authors propose a novel attention based Soft-NER model designed for code and NER. Their BERT model BERToverflow was pre-trained using a cased 64000 WordPiece vocabulary with the same configuration as original BERT\textsubscript{BASE} with 110 million parameters. The results show that BERToverflow clearly outperforms the other models, including BERT\textsubscript{BASE}, on NER tasks. In contrast, we pre-train seBERT with about six times more data, including data from GitHub and Jira. Moreover, seBERT uses an uncased 30522 WordPiece vocabulary and the same configuration as BERT\textsubscript{LARGE}, resulting in a much larger model with 340 million parameters. Additionally, our focus is different and not on NER tasks, but rather on the general validity of the NLP models and their usefulness for classification tasks in a pure NLP setting without considering source code.

While our focus is on pure NLP, we also want to mention similar models from the SE domain for code. Theeten et al. [34] proposed import2vec, a word2vec based approach for learning embeddings for software libraries. The embeddings are trained on import statements extracted from source code of Java, JavaScript and Python open source repositories. In their work, they show that their embeddings capture aspects of semantic similarity between libraries and that they can be clustered by specific domains or platform libraries. A more general source code based pre-training was conducted by Alon et al. [35] who proposed code2vec to represent code snippets as word embeddings. The authors show that such embeddings are a powerful tool to predict method names based on the code. Another source code based approach is CodeBERT [36], a bimodal pre-trained bidirectional transformer for natural language and programming languages. CodeBERT is pre-trained using a hybrid objective function combining the default MLM task and a Replaced Token Detection (RTD) [37] task. For the training data, the authors use both bimodal data pairs comprising a function (code) and its documentation (natural language), as well as unimodal data consisting of only of the function. The data was collected from GitHub code repositories in six different programming languages. Their results show that CodeBERT achieves state-of-the-art performance in natural language code search and code-to-document generation tasks.

4 seBERT

Within this section we describe how we trained seBERT, a large-scale transformer model for the SE domain that complements BERToverflow, because it is larger, was trained on more data diverse data with a larger volume, and is uncased (see Section 3).

4.1 Data

We identified four data sources for our domain-specific corpus of SE textual data:

1) Stack Overflow posts: With millions of questions, answers and comments, Stack Overflow posts are a rich source of textual data from the SE field. Same as the prior work, this is one of our main data sources. The Stack Exchange Data Dump [38] contains Stack Overflow posts from 2014 to 2020 and is hosted on the Internet Archive [39]. A Stack Overflow specific mirror is available as a public dataset on the Google Cloud Platform [40]. Using BigQuery, we extracted 62.8 Gigabyte of questions, answers and comments.

2) GitHub issues: GitHub is more than a code hosting environment. For many users, GitHub issues are the first place to give feedback or report software bugs and thus provide valuable insight into the communication between users and developers. GitHub issues consist of a title, a description and comments and are available through the GitHub Archive [41]. Using Google BigQuery [42] we extracted 118.5 Gigabyte of issue descriptions and comments from the years 2015 to 2019.

3) Jira issues: Similar to GitHub issues, Jira issues provide valuable insights into software team communication regarding bug and issue tracking. In 2015, Ortu et al. published the Jira Repository Dataset [43], which contains 700 K Jira issue reports and more than 2 M Jira issue comments. With 1.4 Gigabyte of unprocessed textual data, they make up the smallest part of our corpus. However, the overall amount of data is still fairly large, and provides a perspective beyond Stack Overflow and GitHub. This includes language regarding the committing solutions to Subversion, or the submission of textual diffs as patches.

4) GitHub commit messages: Git commit messages are used to communicate the context of changes to other developers. Commit messages consist of two parts, a subject line and a message body, the latter being optional. Similar to the GitHub issues, we extracted 21.7 Gigabyte of GitHub commit messages from the GitHub Archive using BigQuery.

In total, our data set consists of 204.4 Gigabyte of unprocessed textual data. To the best of our knowledge, this the largest and most diverse corpus of textual data in the SE domain. Prior work on pre-training focused only on data from Stack Overflow. However, this ignores important aspects of the SE domain. A key aspect of natural language communication is the description of feature requests and bugs. Such data is typically not available on Stack Overflow, with the exception of users asking how they could work around a specific issue. Through the inclusion of GitHub issues and Jira issues, we enhance the corpus with such data. Moreover, commit messages are often short and on point natural language summaries of development activities and, consequently, different from the often longer discussions within issues and on Stack Overflow.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
4.2 Preprocessing

The textual data is mostly unstructured and needs to be preprocessed. Overall, we conducted eight different preprocessing steps.

1) **Basic preprocessing**: We convert all documents to lowercase, remove control characters (newline, carriage return, tab) and normalize quotation marks and whitespaces.

2) **English**: Since we are training a model for the English language, we use the fastText library to remove all non-English documents.

3) **HTML**: We remove HTML tags and extract text using the BeautifulSoup library.

4) **Markdown**: We use regular expressions to greedily remove Markdown formatting.

5) **Hashes**: Hashes such as SHA-1 or md5 do not provide any contextual information and should be removed. We detect hashes by checking whether alphanumeric words with a length of 7 characters or more can be cast to a hexadecimal number and replace them with [HASH] tokens.

6) **Code**: Source code is not natural language and should be removed. However, finding code fragments within text is a non-trivial task. We use HTML <code> tags, Markdown code blocks and other formatting to identify source code and replace it with [CODE] tokens. Code that is not within such environments is not filtered.

7) **User mentions**: For privacy reasons, we replace user names and mentions (@user) with [USER] tokens.

8) **Special formatting**: We remove special formatting and content such as Jira specific formatting, Git sign-off, or references to SVN.

The preprocessing steps differ for the data sources. For example, removal of Markdown does not make sense for Stack Overflow posts or commit messages. Table 1 shows which steps we applied to which data and Table 2 shows the amount of data from each data source after preprocessing. After preprocessing, a total of 119.7 Gigabyte of text or 20.9 billion words remain.

4.3 Pre-Training

The BERT implementation by Devlin et al. [2] uses WordPiece embeddings with a 30522 token vocabulary. We train a new SE domain-specific vocabulary of the same size using all our data and the BertWordPieceTokenizer by HuggingFace.

An important parameter in BERT pre-training is the maximum sequence length. Training sequences shorter than the maximum sequence length are padded with [PAD] tokens, while longer sequences are truncated. Since the self-attention mechanism of BERT has a quadratic complexity with respect to the sequence length [2], the parameter significantly affects the training time and the memory requirements. Therefore, the authors recommend training 90% of the training steps with a sequence length of 128 and 10% of the steps with a sequence length of 512 to learn longer contexts [2]. We analyzed the sequence length for all our data through histograms, which are shown in Fig. 2. The sequence lengths follow exponential distribution and 96.7% of all training sequences are shorter than 256 and 90.1% are shorter than 128. Therefore, since most of our data is short-sequence data, we train with a sequence length of 128 for all steps at the cost of a possible small disadvantage with very long contexts.

## Table 1

| Processing       | GitHub issues | Commit messages | Stack Overflow | Jira issues |
|------------------|---------------|-----------------|----------------|-------------|
| Basic            | X             | X               | X              | X           |
| English          | X             |                 |                |             |
| HTML             | X             | X               | X              |             |
| Markdown         | X             |                 |                |             |
| Hashes           | X             | X               | X              | X           |
| Code             | X             |                 |                | X           |
| User mentions    | X             |                 |                |             |
| Special formatting |               |                 |                |             |

## Table 2

|                  | GitHub       | Stack Overflow | Jira            |
|------------------|--------------|----------------|-----------------|
| Issues           | 29.7 Gigabyte| 10 Gigabyte    | 502 Megabyte    |
| Comments         | 39.3 Gigabyte| 10.1 Gigabyte  | 613 Megabyte    |
| Commit messages  | 18.2 Gigabyte| 11.3 Gigabyte  |                 |

2. https://fasttext.cc/
3. https://beautiful-soup-4.readthedocs.io/
4. https://huggingface.co/
5. https://www.nltk.org/
code from the original BERT to prepare the data for the MLM task. The provided script duplicates the input sequences by a dupe factor and creates training samples by randomly masking 15% of the whole words. Whole word masking was not part of the original BERT implementation and only added later by the authors as improvement of the preprocessing. With whole word masking all tokens of a word are masked at once, which means that it is not possible that words are only partially masked. Fig. 3 shows this for a document with three sentences. As a result of this data preparation, we have 2.4 Terabyte of data which we can use as input for the pre-training of BERT with TensorFlow.

We pre-train seBERT using the same configuration as BERT-LARGE, i.e., with 24 layers, a hidden layer size of 1024, and 16 self-attention heads, which leads to a total of 340 million parameters. Pre-training of BERT is expensive, e.g., the original BERT-LARGE was pre-trained on 16 Cloud TPUs for 3 days. Since then, several optimizations have been proposed to reduce the training time of BERT. You et al. [44] proposed a Layer-wise Adaptive Moments Based (LAMB) optimizer that allows the use of larger minibatches compared to the originally used ADAMW optimizer. NVIDIA then implemented their own version NVLAMB and combined it with Automatic Mixed Precision (AMP) [45], which accelerates training time and reduces memory usage by autocasting variables to 16 b floating point numbers upon retrieval, while storing variables with the usual 32 b precision. We used this NVIDIA implementation of the BERT pre-training to facilitate the effective usage of the system we had available with 2x24 CPU cores, 384 GB RAM and 8x NVIDIA Tesla V100 32 G GPUs. The pre-training of seBERT required about 4.5 days.

5 Validation of SE Domain Models

The evaluation of pre-trained language models is usually performed with benchmarks such as the GLUE benchmark [46], its successor SuperGLUE [47] or SQuAD [48]. However, these benchmarks require fine-tuning and are designed for the general domain and therefore not suitable for the evaluation of NLP models for the SE domain. Moreover, while the evaluation on benchmark data is useful to evaluate the overall predictive capabilities of a model, they are not suitable to understand the reasoning within a model. Therefore, we propose a different approach for the evaluation of transformer models within the SE domain that considers the validity from three different perspectives: the vocabulary, context sensitive prediction of masked words, and a fine-tuned benchmarks using a limited amounts of data for classification.

5.1 Vocabulary

One of the main differences between a domain-specific language model and a general domain language model should be the vocabulary. We expect that the SE specific models should gain words from the SE domain, while lost words should be from other domains. Since alphanumeric words are more prevalent in the domain, a larger proportion of words should be alphanumeric.

We take pattern from Beltagy et al. [6] and perform a vocabulary analysis by computing the WordPiece overlap,
i.e., the percentage of identical tokens between vocabularies. Additionally, we examine the vocabularies for differences in structure, tokenization, and the types of words gained or lost through a manual analysis of the differences. The manual analysis was conducted by two authors who both read the word lists and took notes about differences. The notes were then compared and discussed during a virtual meeting, which resulted in the qualitative assessment of the differences we provide in Section 6.1.

5.2 Contextual Embeddings

Transformer models such as BERT go beyond word similarities and instead consider the complete context to infer the meaning. This goes beyond the context-free finding of similar words that was used by Efstatthiou et al. [9] to determine if word embeddings from the SE domain are valid. Instead, we suggest to use MLM tasks to evaluate the validity of the embeddings, i.e., the evaluation of understanding how good the model is at predicting missing words in a text [2, 6]. We devise three categories of sentences suitable for validating language representation models for the software engineering domain with the goal to understand if seBERT and BERTOverflow behave as expected due to the differences in training data from the original BERT. The three categories are the following.

1) **Positive examples**: Prediction of masked words within a software engineering context, i.e., where the missing word or context words are polysemous or from the SE domain. We expect domain-specific models to produce accurate predictions, while the original BERT may struggle because of the SE context.

2) **Negative examples**: Prediction of masked words outside a software engineering context, where the missing word or context words are polysemous. Given the polysemy of the negative examples, domain-specific models should produce domain-specific predictions, which are not suitable for the sentences from the general domain.

3) **Neutral examples**: Prediction of masked words outside a SE context, where the missing word can be inferred from general language understanding (e.g., idioms, opposites, antimetaboles). General language understanding should be preserved across domains, we expect all models to perform well.

Since sentences of these categories can only demonstrate the validity by way of example, we refer to them as validation examples. We defined ten examples per category. Due to space restrictions, we only list the sentences together with the predictions by the models in Section 6.2 in table 4-6. For each category, we defined ten validation examples. To define these examples, one author created a list of candidate sentences including the word that should be masked and the expectation. These candidates were prepared based on the polysemous terms from Efstatthiou et al. [9]. Then, all authors discussed and refined these candidates together in a virtual meeting to define the ten examples per category. After this process, we can then compare how the different models complete the sentences and how this matches our expectation.

The drawback of our validation examples is that while they provide valuable insights, they are not an unbiased large sample, but rather a small and manually designed sample with a specific purpose. This raises potential issues regarding the generalizability of the effects we find to real world data. We define the following experiment with the goal to compare the differences of contextual embeddings within the SE domain on a large scale. The general idea is to conduct a MLM experiment in which polysemes are masked and then compare how good the models are at predicting the polysemes.

- Determine the overlap between the vocabularies of all involved models and discard all subwords.
- Manually extract a list of polysemes from the overlapping words, i.e., words that have a different meaning in the SE domain and the general domain (e.g., shell, bug, root).
- Collect two large corpora of text data: one from the SE domain and one from the general domain. Both corpora must not have been used during the pretraining of the models. In our case, we collected the Stack Overflow posts from 2021 as SE domain corpus (2.9 GB) and used the CC-News corpus (2.6 GB)7 as general domain corpus.
- For each of the polysemes, randomly sample 100 sentences from both corpora. If there are less than 100 such sentences, drop the polysemes from the analysis.
- Mask the polyseme in each sentence and query the model for the five most likely words.

We can then evaluate the different models with respect to their capability to find the masked word. We would expect that models from the SE domain perform better on the Stack Overflow corpus and, vice versa, models from the general domain perform better on the CC-News corpus.

5.3 Fine-Tuned Prediction Tasks

The first check regarding the validity is to analyze the pretrained model directly. However, it is unclear if the actual use of these models in fine-tuned applications would also be affected. We propose to evaluate the validity of the models by applying them to a classification task within the SE domain, for which only a limited amount of data is available. We propose to always use (at least) three different tasks to evaluate the capabilities of the fine-tuning.

- The first type of task should have longer texts. This task is suitable to evaluate how good the model is at understanding longer texts and contexts.
- The second type of task should have shorter text, to evaluate how good the model is for short texts.
- The third task should be sentiment mining, as a general domain use case that is already well understood in the SE domain [49], incl. strong, already existing results that general domain transformer models are the current state-of-the-art for sentiment mining in the SE domain [14]. Since the work by Novielli et al. [49] suggests that SE domain models for sentiment mining do not yet work very well because their results are unstable, and Novielli et al. [50], Lin et al.

---

7. https://huggingface.co/datasets/cc_news
[51], and Zhang et al. [14] showed that smaller general domain models frequently outperform SE-specific models. This task allows us to evaluate if pre-trained SE domain transformer models are also better on SE domain data for use cases that may not require domain knowledge.

For all fine-tuning tasks, we propose to simply re-use the classification implementation for BERT from HuggingFace.9 this implementation adds a single fully connected layer which produces the classification based on the output sequence of the BERT model. We then train each model for five epochs on the issue, commit, respectively, sentiment data for the fine-tuning. We use 80% of the training data for the fine-tuning and 20% as validation data to determine when the models start to overfit.

We note that we use different fine-tuning tasks than the NER task that was used by Tabassum et al. [10]. Our rationale for using a classification task instead is that the NER task is very specific, especially with the respect that the exploitation of casing and special characters like the underscore is a major aspect of NER within natural language models. Thus, this task does not aim to understand the meaning, but rather the inference of a specific language construct. In comparison, our tasks require the interpretation of the meaning of natural language.

In the following, we discuss how we implement and evaluate each of the fine tuning tasks in detail.

### 5.3.1 Prediction of Bug Issues

We propose to use the prediction if issues are bugs [11] as task with longer sequences. This task has several properties that make it a good candidate to evaluate the validity and benefits of transformer models. Issues are among the longer texts within SE data (see Fig. 2). Moreover, there is an established baseline performance which shows that a simple neural network approach without pre-training based on fastText [52] with built-in automated hyperparameter tuning performs well [11]. Thus, we can not only compare pre-trained models to each other, but also their benefit over simpler models that can be trained in minutes without any special hardware. Second, the task is hard and often not solved correctly by humans, unless careful manual validation is used [12], [53], [54].

We use the five projects with validated issue type data from from Herzig et al. [12] together with the 38 validated projects from Herbold et al. [54] together as a single data set with 43 projects and conduct a leave-one-project-out cross validation experiment. This means that one project is used for testing and the remaining 42 projects for the training of the classifier, i.e., of the fastText model as baseline and for the fine-tuning of the BERT models. Since there are 38,219 issues in the data and the largest project has 2,399 issues, we create 43 fine-tuned models trained with at least 35,820 issue descriptions. Following Herbold et al. [11], we conduct a statistical comparison based on the F1 score which is defined as

\[
\text{recall} = \frac{tp}{tp + fn}
\]

\[
\text{precision} = \frac{tp}{tp + fp}
\]

\[
F1\text{score} = 2 \cdot \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}
\]

where \( tp \) are true positives, i.e., bugs that are classified as bugs, \( tn \) true negatives, i.e., non bugs classified as non bugs, \( fp \) bugs not classified as bugs and \( fn \) non bugs classified as bugs. Additionally, the recall and precision should be reported to allow us to understand the nature of the errors made by the models. We use a Bayesian approach for the statistical comparison following the guidelines by Benavoli et al. [55]. The authors suggest to use a Bayesian signed rank test for a comparison of results on multiple data sets, as is the case here. The advantage of the Bayesian approach is that the number of models that are compared does not influence the results, as this directly estimates that one approach performs better, which does not require an adjustment of the significance level [55].

### 5.3.2 Commit Intent Classification

As second fine-tuning task with shorter sequences, we use the automated classification of commits based on commit messages. Commit messages are usually very short (see Fig. 2). Unfortunately, we are not aware of a solid benchmark for the machine-learning based classification of commit messages. Due to the good results for issues, we assume that fastText is also a good baseline for the classification of commit messages. As data, we use manually validated data from Trautsch et al. [13], i.e., we have a sample with a total size of 2533 commit messages that has as class label whether a commit is perfective or not. We follow the approach outlined by Benavoli et al. [55] for the comparison of classifiers on a single data set: we conduct 10x10 cross-validation and obtain 100 results for each classifier. We then use the Bayesian correlated t-test to compare the results. Same as above, we use the F1 score as foundation for the statistical comparison and report the recall and precision to understand the nature of the errors made by the models.

### 5.3.3 Sentiment Mining

As third fine-tuning task we propose sentiment mining [14], [49]. We use three data sets that were also used by Zhang et al. [14] in a prior benchmark: the SO [51] and API [56] data with 4522 resp. 1500 sentences from Stack Overflow posts and the GH data [57] with 7122 sentences from GitHub pull requests and commit messages. All three data sets have three labels, i.e., positive, neutral, and negative sentiment. For each of these data sets, we use the same experimental setting as for the second fine-tuning task, i.e., we conduct 10x10 cross-validation. Since Zhang et al. [14] also the F1 score, we follow this approach and report the macro averages of the F1 score, recall, and precision, same as for the other use cases.
**6 Experiments**

We conducted an experiment to evaluate the validity of the SE models seBERT and BERToverflow in comparison to the general domain BERT models. Since seBERT is based on BERT\_LARGE and BERT\_BASE, we use both of these models within our comparison. We follow the procedure outlined in Section 5. This means we first compare the vocabularies. Then, we proceed to look at the validity of the models without fine-tuning through their ability to infer the correct meaning of terms. Finally, we evaluated how the different models perform when fine-tuned with a limited amount of labeled data for a prediction task.

All results, as well as the required scripts and link to the data for pre-training seBERT are available as part of our replication kit. Additionally, we prepared a playground, which can be used to fill in masked words using the models.

We restrict our comparison to BERT, as this is the most similar general domain model and, therefore, most suitable to explore the need for SE-specific models for SE domain tasks. Please note that we also exclude CodeBERT from this comparison, because CodeBERT is not a natural language model, but rather a bi-modal “natural language - programming language” model, i.e., for working with source code.

Additionally, we use fastText for the comparison with respect to classification tasks, to understand if smaller NLP models may be sufficient.

**6.1 Vocabulary Comparison**

The vocabularies of BERT\_BASE and BERT\_LARGE are equal, and we refer to them as BERT vocabulary in the following. The WordPiece overlap between BERT and seBERT is 38.3%. A major difference between the vocabularies is the number of “##” sub-word WordPieces, with seBERT (7430) having almost twice as many as BERT (3285). Most of the words added to the seBERT vocabulary are from the SE domain (e.g., “bugzilla,” “jvm,” “debug”) or internet slang (e.g., “fanboy”). Lost words are mainly from the geographical (e.g., “madrid,” “Switzerland,” “Egypt”), religious (e.g., “jesus,” “buddha”) or political (e.g., “president,” “minister,” “clinton”) domain.

For words that occur in only one of the two vocabularies, we use the tokenizer of the other model to break them down into their respective WordPieces. We have presented the results for a small subset of words in Table 3. As we can see, BERT tokenizes SE domain words inconsistently, e.g., “bugzilla” is tokenized into “bug ##zil ##la,” but “debug” into “de ##bu ##g,” showing that there is no WordPiece for “##bug”. In addition, in-domain abbreviations such as JVM are unknown and broken down into their individual characters. In contrast, seBERT breaks down general out-of-domain words into in-domain WordPieces, e.g., “drama” into dynamic random-access memory (DRAM) or “infantry” into “inf” and “ant”. Since the vocabularies are frequency-based, an interesting finding is that “woman” is not in the seBERT vocabulary and is tokenized into “wo ##man”. This implies that “woman” is used less frequently within the domain, highlighting the ethical issues that can arise in machine learning. The full lists of out-of-vocabulary words and their tokenizations are available as supplemental material, in the seBERT replication repository.

Since the BERToverflow vocabulary is cased, a direct comparison of the overlap without further processing is not feasible. Instead, we lower case the vocabulary of BERToverflow and remove duplicates. Since the vocabulary of BERToverflow is larger than that of (se)BERT, we cannot directly compute the overlap, because this is a ratio with respect to the vocabulary size. The larger BERToverflow should contain more words from the other vocabularies (larger ratio), while in turn the smaller the (se)BERT vocabularies should cover a smaller ratio of terms in the BERToverflow vocabulary. Instead, we calculate i) the ratio of WordPieces of the (se)BERT vocabulary within the uncased BERToverflow vocabulary and ii) the ratio of uncased BERToverflow WordPieces within the BERT vocabulary. Thus, consider the direction of the comparison, when considering ratios.

After lower casing, there are 58,854 unique WordPiece tokens. 24,263 of these tokens are also in the seBERT vocabulary. This means seBERT covers about 40% of the uncased BERToverflow vocabulary and the uncased BERToverflow covers about 80% of the seBERT vocabulary. That BERToverflow has more tokens is not surprising, due to the larger vocabulary size. While the overlap of 80% of the seBERT tokens with BERToverflow is substantial, we would have...
expected an even larger overlap, given that BERToverflow has about twice the amount of uncased tokens. This should be sufficient to more or less cover the seBERT vocabulary, assuming that the textual data from Stack Overflow on which BERToverflow is pre-trained, is representative for the SE domain. A manual check revealed that many of the missing tokens are related to tools, e.g., “matplot”, “xenial” or “hashicorp”. We can only speculate regarding the reason for this. One possible explanation is that tools are less frequently discussed on Stack Overflow than, e.g., programming languages. However, while this may the case to a certain degree, questions regarding the usage of specific tools are common in Stack Overflow. Another possible explanation is that the casing that is used when writing tools. For example, users may write “HashiCorp,” “Hashicorp,” or “hashicorp”. Within the uncased seBERT vocabulary, this would not make a difference and all occurrences would be counted together, possibly leading to larger importance and the inclusion in the vocabulary.

We also manually evaluated which additional terms are within the BERToverflow vocabulary that are not within the seBERT vocabulary. In addition to more terms from the general domain (see the comparison to BERToverflow below), we noticed two additional aspects. First, there were many Unicode tokens, such as special characters only used in non-English texts. Second, there were many tokens that seem to come from code snippets, such as “strptime”.

The BERT vocabulary has an overlap of 15,926 WordPiece tokens with the uncased BERToverflow vocabulary. This means that about 50% of the BERT tokens are within the BERToverflow vocabulary, which is slightly more than the overlap of 38% between seBERT and BERT. However, this increase is plausible, given the overall larger vocabulary size of BERToverflow. This is also visible in the tokenization, e.g., the terms “women” and “drama” are within BERToverflow vocabulary, meaning that it covers more general domain concepts that seBERT.

6.2 Contextual Comparison

Next, we use the capability of the models to predict masked words. Tables 4-6 show ten examples of sentences for the positive, negative, and neutral category.

The positive examples in Table 4 show that while the general domain BERT models are not as accurate with the identification of words from the within-domain context as the in-domain models BERToverflow and seBERT. This does not mean that BERT outright fails, but rather that the inferred are not completely unrelated, but also not directly on point. In comparison, BERToverflow and seBERT both almost always suggest reasonable completions. However, we also observe that either the larger set of training data or the larger model also allows seBERT to be more accurate in some cases. Our sensitivity analysis with a smaller version of seBERT shows that likely both play a role, as the smaller model is almost equal, but worse in some nuances, i.e., handling of emoticons and opposites. The sensitivity analysis can be found in the supplemental material, available online. The last example gives the strongest indication of the impact of using more data: since BERToverflow was not trained on issue data, but only on Q&A data from Stack Overflow, seBERT performs better in this context. For this sentence, the general domain results from BERT are actually better than BERToverflow. Our interpretation of this is that this is caused by the context of the BERToverflow data: the criticality of issues is usually not discussed on Stack Overflow. Instead, users may ask regarding a certain problem, with other users responding that the bug is known. Thus, the association of “This is a [MASK] bug” with “known” makes sense. However, the second part of the sentence further clarifies this context as “please address this asap”. Such a direct request to the developers does not make sense within the Q&A context and could, therefore, not be considered by BERToverflow. In comparison, the seBERT data is aware of such direct communication with a development team and correctly infers that the missing word is likely the criticality of the issue, because it should be addressed as soon as possible.

The negative examples in Table 5 highlight that models trained with SE data fail, when it comes to understanding pure general domain context. All words are interpreted within the SE context, even if this does not make sense. In comparison, the general domain models were a lot closer to the performance of the SE domain models for the positive examples. When we consider the training corpus, this makes sense: topics like dentists, actual snakes, or the weather are extremely unlikely in our SE corpus, so it is plausible that this does not work at all. On the other hand, Wikipedia also covers software engineering topics. Thus, while this is not the focus of the general domain corpus, it also contains some software engineering data.

The neutral examples in Table 6 reveal some nuanced differences between all four models. This is the first time that we clearly observe that the BERTLARGE model is better a capturing the context than the BERTBASE model. In Sentence 28, the smaller models fail to correctly understand the context, while all other models understand this perfectly. The same sentence also reveals a nuanced difference: since we often start to count from zero in computer science, this is also proposed by BERToverflow and seBERT, while BERTLARGE says to start at one. We also note that the SE-specific models sometimes misjudge these neutral concepts, e.g., with sentence 23. While the sentence clearly indicates that the context is voting, the term “raise” is so strongly associated with opinions within the SE domain that hands are not suggested. This set of sentences also contains the strangest association within our data in Sentence 30: we have no idea why seBERT believes that running out of pokemon is likely. The only explanation we can think of is the tokenization by seBERT of “poker” as “poke##r”. It is possible that the subword “poke” has a strong association with “pokemon,” which leads to this mistake.

Fig. 4 shows the results of our experiments of masking polysemes in the Stack Overflow corpus and CC-News corpus. Overall, we identified 503 words, which appear in at least 100 sentences in each corpus. Moreover, the results are in line with our expectation: seBERT and BERToverflow are better for the Stack Overflow corpus, the BERT models are better for CC-News. The full list of polysemes can be found in the supplemental material, available online.
larger models (seBERT, BERT\_LARGE) perform a bit better than the smaller models (BERToverflow, BERT\_BASE). An interesting aspect of the models is that the results are symmetric: the seBERT results on Stack Overflow are similar to the BERT\_LARGE results on CC-News and the seBERT results on CC-News are similar to the BERT\_LARGE results on Stack Overflow. The same relationship can be observed between BERToverflow and BERT\_BASE. This could mean that the contextual understanding of the SE domain models for the general domain is about as good as that of the general domain models for the SE domain, and vice versa.

6.3 Prediction Task Comparison

Table 7 and Figs. 5, 6, 7, 8, and 9 summarize the results of the prediction task comparison. In the following, we look at the results of the individual tasks in detail.

| Sentence | Expectation | BERT\_BASE | Probability | BERT\_LARGE | Probability | BERToverflow | Probability | seBERT | Probability |
|----------|-------------|------------|-------------|-------------|-------------|--------------|-------------|---------|-------------|
| 1) The [MASK] is thrown when an application attempts to use null in a case where an object is required. | NullPointer-Exception | rule | 0.2407 | value | 0.2356 | exception | 0.4718 | exception | 0.5473 |
| | exception | 0.0742 | exception | 0.0804 | error | 0.1229 | error | 0.2188 |
| | coin | 0.0659 | coin | 0.0342 | Null-Pointer-Exception | 0.1161 | null-pointer-exception | 0.1158 |
| | flag | 0.0245 | ball | 0.0318 | NPE | 0.0995 | illegal-argument-exception | 0.0291 |
| | penalty | 0.0216 | flag | 0.0312 | Illegal-Argument-Exception | 0.0403 | null-pointer-exception | 0.0144 |

| 2) [MASK] is a proprietary issue tracking product developed by Atlassian that allows bug tracking and agile project management. | Jira | it | 0.0382 | it | 0.0148 | Jira | 0.4914 | jira | 0.4954 |
| | agile | 0.0104 | bug | 0.0074 | Redmine | 0.1834 | there | 0.0837 |
| | eclipse | 0.0041 | eclipse | 0.0052 | JIRA | 0.099 | zenhub | 0.0763 |
| | flex | 0.0035 | echo | 0.0046 | It | 0.0398 | it | 0.0488 |
| | snap | 0.0031 | spark | 0.0035 | There | 0.0281 | bugzilla | 0.0463 |

| 3) [MASK] is a software tool for automating software build processes. | build | it | 0.1437 | it | 0.0396 | CMake | 0.1541 | jenkins | 0.1907 |
| | automation | agile | 0.0071 | build | 0.0294 | make | 0.0898 | make | 0.0891 |
| | tools | this | 0.0062 | eclipse | 0.0107 | Make | 0.0644 | r讦江 | 0.0676 |
| | build | 0.0035 | builder | 0.0075 | Ant | 0.0595 | it | 0.058 |
| | gem | 0.0035 | agicle | 0.0067 | Maven | 0.0558 | ant | 0.0536 |

| 4) Pathlib is a python library used for handling [MASK]. | paths | applications | 0.0699 | applications | 0.1509 | paths | 0.429 | paths | 0.8727 |
| | systems | 0.0431 | software | 0.1505 | files | 0.0789 | path | 0.0338 |
| | programs | 0.0359 | programs | 0.078 | urls | 0.0664 | directories | 0.0237 |
| | software | 0.0351 | languages | 0.0631 | URLs | 0.0391 | filenames | 0.0168 |
| | data | 0.0296 | systems | 0.052 | pathnames | 0.0384 | strings | 0.0088 |

| 5) The solution posted by [USER] is [MASK] helpful. :; | positive adverb | very | 0.5632 | very | 0.3407 | very | 0.5488 | very | 0.5719 |
| | extremely | 0.0656 | not | 0.1755 | really | 0.1374 | really | 0.1533 |
| | quite | 0.0424 | always | 0.0624 | also | 0.0734 | quite | 0.0729 |
| | always | 0.0313 | also | 0.0593 | more | 0.027 | also | 0.0458 |
| | more | 0.0273 | most | 0.0429 | quite | 0.0237 | super | 0.0166 |

| 6) The solution posted by [USER] is [MASK] helpful. ; | negative adverb | very | 0.5254 | very | 0.2764 | not | 0.9182 | not | 0.9761 |
| | extremely | 0.0659 | not | 0.1767 | very | 0.0331 | no | 0.0075 |
| | quite | 0.0418 | also | 0.0829 | really | 0.0087 | very | 0.0038 |
| | also | 0.0349 | always | 0.0782 | never | 0.0031 | really | 0.0011 |
| | always | 0.0338 | most | 0.0455 | also | 0.0031 | never | 0.0009 |

| 7) [MASK], is a provider of Internet hosting for software development and version control using Git. | github, gilab | microsoft | 0.0158 | net | 0.0127 | Github | 0.222 | github | 0.3096 |
| | net | 0.0136 | apache | 0.011 | GitHub | 0.2186 | githlab | 0.0397 |
| | oracle | 0.0134 | parallels | 0.0097 | Bitbucket | 0.0568 | git | 0.0382 |
| | org | 0.0127 | foundry | 0.0073 | Bitbucket | 0.0509 | sourceforge | 0.0309 |
| | apache | 0.0122 | radius | 0.0061 | Assembla | 0.0413 | bitbucket | 0.0302 |

| 8) In object-oriented programming, a [MASK] is an extensible program-code-template for creating objects. | class | template | 0.1027 | template | 0.6763 | constructor | 0.3849 | class | 0.4991 |
| | class | 0.0391 | prototype | 0.0533 | class | 0.2664 | constructor | 0.221 |
| | object | 0.0356 | module | 0.0258 | factory | 0.1435 | factory | 0.0669 |
| | model | 0.0219 | class | 0.0136 | prototype | 0.038 | metaclass | 0.0487 |
| | gui | 0.0145 | construct | 0.0106 | Factory | 0.0182 | template | 0.0316 |

| 9) I have to discuss this with the other [MASK]. | developers | elders | 0.0703 | men | 0.0677 | developers | 0.0918 | maintainers | 0.0797 |
| | girls | 0.0622 | elders | 0.0399 | developers | 0.0918 | maintainers | 0.0797 |
| | officers | 0.0526 | officers | 0.0265 | people | 0.0533 | people | 0.0796 |
| | council | 0.0453 | people | 0.024 | users | 0.0497 | devs | 0.0744 |

| 10) This is a [MASK] bug, please address it asap. | critical, major | serious | 0.286 | new | 0.1166 | serious | 0.0807 | new | 0.287 |
| | new | 0.1166 | bug | 0.0397 | browser | 0.014 | bug | 0.0270 |
| | major | 0.0699 | minor | 0.0458 | chrome | 0.02 | major | 0.0412 |
| | persistent | 0.0473 | major | 0.0397 | browser | 0.014 | bug | 0.0270 |
| | bug | 0.031 | nasty | 0.0298 | common | 0.0118 | real | 0.0099 |

Results for the positive category, i.e., sentences where we expect that BERToverflow and seBERT perform better than BERT.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
6.3.1 Issue Type Prediction

The results show that seBERT and BERToverflow achieve the best performance for the issue type prediction tasks, outperforming both fastText and the general-domain BERT models. The improvement over fastText is very large with an about 11% higher F1 score for the issue type prediction. The violin plot in Fig. 5 indicates that the performance improvement in F1 score is due to an improvement of both recall and precision, which means the models reduced both false positives and false negatives in comparison to fastText. The Bayesian signed rank test determined that this improvement of the SE-specific models over the other models is significant.15 The difference between seBERT and BERToverflow is not significant. The comparison between fastText and BERT_BASE shows that the general-domain models may be better than smaller text processing models without pre-training on the issue type prediction. BERT_BASE significantly outperforms fastText on the issue type prediction task. The BERT_LARGE model

15. Posterior probabilities determined by the Bayesian signed rank test can be found in the supplemental material, available online.

| Sentence | Expectation | BERT_BASE Prediction | BERT_LARGE Prediction | BERToverflow Prediction | seBERT Prediction |
|----------|-------------|----------------------|-----------------------|------------------------|--------------------|
| 11) A [MASK] crawled across her leg, and she swept it away. | bug | 0.1417 | 0.2087 | 0.1068 | 0.1042 |
|         | tear | 0.1277 | spider | 0.1676 | car | 0.0561 | person | 0.0518 |
|         | hand | 0.0841 | tear | 0.1463 | dog | 0.0559 | friend | 0.0472 |
|         | bug | 0.0834 | fleas | 0.048 | bird | 0.0405 | girl | 0.0299 |
|         | mosquito | 0.0534 | fly | 0.0447 | fox | 0.0397 | monkey | 0.0248 |
| 12) Can you open the window, [MASK], please? It’s hot in here. | door | 0.7435 | 0.8776 | link | 0.3264 | pr | 0.2829 |
|         | window | 0.1227 | window | 0.0619 | file | 0.0699 | issue | 0.1386 |
|         | windows | 0.0349 | fridge | 0.0038 | site | 0.0411 | door | 0.0076 |
|         | blinds | 0.0171 | doors | 0.0037 | page | 0.0395 | file | 0.0466 |
|         | curtains | 0.0113 | gate | 0.0035 | url | 0.0307 | link | 0.0352 |
| 13) The reticulated python is among the few [MASK] that prey on humans. | snakes | 0.6433 | snakes | 0.803 | languages | 0.7376 | things | 0.5227 |
|         | species | 0.1645 | reptiles | 0.0779 | things | 0.0836 | languages | 0.1625 |
|         | reptiles | 0.1055 | animals | 0.0696 | tools | 0.0087 | tools | 0.0632 |
|         | animals | 0.0278 | species | 0.0223 | people | 0.0084 | packages | 0.0348 |
|         | lizards | 0.0269 | mammals | 0.0075 | programmers | 0.0073 | programs | 0.0255 |
| 14) “I have a [MASK] request for you.” He said to the waiter. | special | 0.4258 | special | 0.8448 | new | 0.115 | pull | 0.7016 |
|         | business | 0.0859 | specific | 0.0198 | special | 0.0361 | feature | 0.0432 |
|         | personal | 0.0809 | new | 0.017 | test | 0.0257 | change | 0.0171 |
|         | new | 0.0318 | small | 0.0166 | friend | 0.0224 | similar | 0.0146 |
|         | specific | 0.0225 | simple | 0.0128 | support | 0.0213 | merge | 0.0139 |
| 15) He was admitting to a [MASK] he didn’t commit, knowing it was somebody else who did it. | crime | 0.7861 | crime | 0.9565 | commit | 0.1497 | change | 0.2475 |
|         | murder | 0.1162 | murder | 0.0338 | change | 0.1223 | commit | 0.0974 |
|         | sin | 0.0484 | sin | 0.035 | mistake | 0.0844 | fix | 0.0833 |
|         | lie | 0.0084 | felony | 0.0008 | fact | 0.0629 | file | 0.0464 |
|         | suicide | 0.0056 | lie | 0.0005 | file | 0.044 | code | 0.0372 |
| 16) It’s an incurable, terminal [MASK]. | disease | 0.4208 | disease | 0.8134 | error | 0.0777 | issue | 0.0605 |
|         | condition | 0.3208 | illness | 0.0879 | operation | 0.0639 | problem | 0.0523 |
|         | illness | 0.125 | condition | 0.0579 | command | 0.0517 | bug | 0.0443 |
|         | death | 0.0201 | cancer | 0.0203 | problem | 0.0336 | character | 0.0377 |
|         | cancer | 0.0193 | disorder | 0.0049 | ) | 0.0288 | effect | 0.0273 |
| 17) The dentist said I need to have a root [MASK]. | canal | 0.8847 | canal | 0.9827 | account | 0.1919 | password | 0.1476 |
|         | beer | 0.0307 | beer | 0.01 | certificate | 0.0959 | user | 0.1431 |
|         | problem | 0.0167 | stop | 0.0009 | user | 0.0825 | account | 0.089 |
|         | out | 0.005 | break | 0.0005 | node | 0.0588 | access | 0.0522 |
|         | cellular | 0.003 | problem | 0.0005 | access | 0.0474 | certificate | 0.0437 |
| 18) Everything was covered with a fine layer of [MASK]. | dust | 0.4895 | dust | 0.7656 | transparency | 0.0518 | abstraction | 0.1231 |
|         | dirt | 0.1035 | snow | 0.0813 | abstraction | 0.0417 | coverage | 0.095 |
|         | snow | 0.0625 | dirt | 0.0283 | code | 0.0408 | testing | 0.0894 |
|         | paint | 0.0432 | ice | 0.0199 | complexity | 0.0323 | detail | 0.0671 |
|         | powder | 0.01 | sand | 0.0122 | confidence | 0.032 | documentation | 0.0636 |
| 19) There was not a single cloud in the [MASK]. | sky | 0.9009 | sky | 0.9425 | list | 0.0797 | database | 0.0686 |
|         | air | 0.0748 | heavens | 0.0104 | cloud | 0.0361 | list | 0.0663 |
|         | distance | 0.003 | distance | 0.0076 | world | 0.0333 | dataset | 0.0391 |
|         | skies | 0.0021 | air | 0.0041 | center | 0.027 | file | 0.0358 |
|         | room | 0.0017 | east | 0.0032 | database | 0.0263 | cloud | 0.0337 |
| 20) What does it say in your [MASK] cookie? | fortune | 0.4846 | fortune | 0.295 | browser | 0.1262 | session | 0.4255 |
|         | chocolate | 0.0354 | favorite | 0.2597 | session | 0.083 | browser | 0.0994 |
|         | little | 0.0289 | next | 0.0484 | firefox | 0.07 | auth | 0.0572 |
|         | next | 0.0207 | chocolate | 0.0284 | firefox | 0.0508 | cookie | 0.0318 |
|         | sugar | 0.0206 | sugar | 0.0261 | debug | 0.032 | login | 0.0195 |

Results for the negative category, i.e., sentences where we expect that BERToverflow and seBERT perform worse than BERT.
from the general domain has severe problems with this use case, i.e., there are several cases where the models completely failed. This happened with none of the other models and may be an indication that the amount of data is too small to fine-tune such a large model from the general domain on a domain-specific corpus.

### 6.3.2 Commit Intent Prediction

The results for the commit intent prediction are mostly in line with the results for the issue type prediction. seBERT has a 9% larger F1 score than the non-SE models, BERToverflow is 6% better. The violin plots in Fig. 6 also indicate a stable improvement in both recall and precision and the Bayesian signed rank test determined that the improvement is significant. In comparison to the issue type prediction, seBERT is significantly better than BERToverflow with an absolute difference of about 3% in the F1 score. Moreover, we note that the general domain BERT models fail to outperform fastText and that BERTLARGE has the same stability issues as for the issue type prediction.

| Sentence | Expectation | BERT\textsubscript{BASE} Prediction | Prob. | BERT\textsubscript{LARGE} Prediction | Prob. | BERToverflow Prediction | Prob. | seBERT Prediction | Prob. |
|----------|-------------|-------------------------------------|-------|-------------------------------------|-------|-------------------------|-------|------------------|-------|
| 21) We can [MASK] in person if you have any specific questions. | meet | talk | 0.4357 | meet | 0.695 | help | 0.2649 | chat | 0.5492 |
| 22) [MASK] its name, vitamin D is not a vitamin. Instead, it is a hormone that promotes the absorption of calcium in the body. | despite | whatever | 0.999 | despite | 0.998 | Despite | 0.7482 | despite | 0.9396 |
| 23) Would all those in favour please raise their [MASK]? | hands | hands | 0.5325 | hands | 0.537 | opinion | 0.2422 | opinion | 0.1235 |
| 24) She surprised him with a [MASK]. | smile | positive | 0.4748 | smile | 0.4086 | surprise | 0.1815 | bug | 0.0604 |
| 25) Whoever is happy will make others happy [MASK] too. | happy | happy | 0.898 | happy | 0.9716 | happy | 0.9686 | | |
| 26) If there are night day owls are there [MASK] owls too? | night | day | 0.3312 | day | 0.5414 | day | 0.1331 | day | 0.6346 |
| 27) Never forget, remember always remember. Always forget, never [MASK]. | forget | remember | 0.979 | forget | 0.9327 | remember | 0.4836 | remember | 0.881 |
| 28) If you are counting things, start from [MASK]. | scratch | one | 0.5561 | one | 0.1389 | 0 | 0.3938 | zero | 0.34 |
| 29) Soccer has really rocket simple rules. It’s not [MASK] science. | a | rocket | 0.7412 | rocket | 0.9125 | rocket | 0.8597 | computer | 0.4413 |
| 30) He ran out of money, [MASK], so he had to stop playing poker. | money | time | 0.8096 | money | 0.7833 | cards | 0.4876 | pokemon | 0.1655 |

Results for the neutral category, i.e., sentences where we expect that BERToverflow and seBERT perform worse than BERT.
6.3.3 Sentiment Mining

The results for the sentiment mining show that there are only small differences between the SE domain and general domain BERT models on all three data sets. The absolute performance on the API and GH data is almost equal. The difference on the SO data is slightly larger, where BERToverflow and seBERT outperform the BERTBASE and BERTLARGE by about 3%, with a statistically significant difference. We observe that, same as before, the BERTLARGE model is sometimes unstable, likely for the same reasons as above. The difference between the transformer models and fastText is huge for this task, i.e., at least 20% in F1 score on all data sets.

7 Discussion

We now discuss our results with respect to our research questions, consider the ethical implications of our work, discuss the limitations and open issues, as well as the threats to the validity.

7.1 Interpretation of SE Terminology

Our results indicate that SE domain models are better at modeling natural language within the SE domain than general domain models. The vocabularies contain more tokens from the SE domain. Especially the focus of the vocabulary is interesting. Both seBERT and BERToverflow have the names of many tools such as programming languages, libraries, build tools, and operating systems. Such words are mostly missing in the BERT vocabulary.

That domain specific words replace terms from other domains is not surprising. However, the magnitude of the difference between the transformer models and fastText is huge for this task, i.e., at least 20% in F1 score on all data sets.

Fig. 4. Results of the MLM experiment on unknown data. The histograms show the position of the masked word in the top five predicted words of each model, or if the word was missed.

Table 7

| Model       | Issue Type | Commit Intent | Sentiment (SO) | Sentiment (API) | Sentiment (GH) |
|-------------|------------|---------------|----------------|----------------|----------------|
| fastText    | 0.69       | 0.75          | 0.44           | 0.37           | 0.47           |
| BERTBASE    | 0.77       | 0.75          | 0.73           | 0.57           | 0.91           |
| BERTLARGE   | 0.62       | 0.71          | 0.73           | 0.55           | 0.91           |
| BERToverflow| 0.81       | 0.81          | 0.77           | 0.58           | 0.92           |
| seBERT      | 0.80       | 0.84          | 0.76           | 0.57           | 0.92           |

Macro-average over the three classes for the sentiment mining.

Fig. 5. Results of the prediction of bug issues.

6.3.3 Sentiment Mining

The results for the sentiment mining show that there are only small differences between the SE domain and general domain BERT models on all three data sets. The absolute performance on the API and GH data is almost equal. The difference on the SO data is slightly larger, where BERToverflow and seBERT outperform the BERTBASE and BERTLARGE by about 3%, with a statistically significant difference. We observe that, same as before, the BERTLARGE model is sometimes unstable, likely for the same reasons as above. The difference between the transformer models and fastText is huge for this task, i.e., at least 20% in F1 score on all data sets.
vocabularies require exact matches of word pieces, while the lexical similarity requires only a “similarity in both form and meaning” [59], this highlights how different texts from the SE domain are from the general domain. This also demonstrates that NLP models with a fixed vocabulary should always be retrained from scratch for the SE domain to maximize the performance, instead of basing them on general domain models with additional pre-training steps, as is, e.g., done for BioBERT [7] and ClinicalBERT [8].

The MLM task demonstrates that the better representation of the SE context goes beyond the vocabulary. With SE domain statements, the general domain models often understand that they should suggest a word from the SE domain, but often do not really understand the exact meaning, which leads to unsuitable suggestions. The SE-specific models are much better at understanding the complete context of the missing word and provide suitable suggestion. However, the examples also show that for a SE domain NLP model, data from Stack Overflow alone is not sufficient, as this only represents the domain within a Q&A context. At least some other aspects of the NLP aspects of the SE domain are not sufficiently captured by the BERToverflow model within our examples. E.g., the lack of issue handling on Stack Overflow translated into a problem with identifying the correct meaning of a missing term. This demonstrates that a larger amount of training data is beneficial and that data should be selected from a diverse range of sources. This also shows that we should be careful, when we use NLP models for tasks, where the text may be different from the pre-training texts. The additional MLM examples confirm what we

![Fig. 6. Results of the prediction of quality improving commits.](image1)

![Fig. 7. Results of the sentiment mining on the SO data.](image2)

![Fig. 8. Results of the sentiment mining on the API data.](image3)
expected about the SE domain models: their general language understanding is okay, but they are clearly inferior to general domain models for anything beyond the SE domain.

**Answer to RQ1:** Transformer models trained with general domain data have trouble understanding SE domain terminology. SE specific models are very good at understanding sentences within the SE context and in general language understanding, but do not perform well in contexts beyond the SE domain.

### 7.2 Impact on Applications

The results of the fine-tuned predictions show that SE domain transformer models easily are state-of-the-art for the two domain specific NLP tasks we considered, i.e., the issue type prediction and the commit intent prediction. In both tasks, seBERT and BERT over flow clearly outperform the competition and the absolute improvement is huge. Even more importantly, these models are decreasing both false negatives and false positives, i.e., they do not achieve this performance improvement through a trade-off between the errors, which indicates that this is really due to a better understanding of the language. Thus, the high validity of the domain understanding we found through the manual inspection of the vocabularies and the MLM task also translates into a better performance in relevant NLP use cases. This improvement is not explainable by the model size alone. This is demonstrated by the lower performance of the general domain BERT models. These models have the same size as our in-domain models\(^\text{16}\), but do not show the same improvement over the smaller fastText model.

Moreover, our results show that the SE domain transformers are not always better when SE data is used: we could not find notable differences between the general domain BERT models and the SE domain models for the sentiment mining tasks. Thus, SE specific pre-training only seems valuable, if the use case is specific to the SE domain and requires the interpretation of text with a SE background, which based on recent research does not seem to be the case for sentiment mining [49].

\[\text{Answer to RQ2:}\] The impact of our choice to use a sequence length of only 128, in comparison to the sequence lengths of 512 from the other models is also visible in the fine-tuning. For the short commit messages of the commit intent prediction task, the seBERT model outperforms all other models, including BERT over flow. This is reasonable, because of the seBERT is the overall larger model and commit messages were used for the pre-training. We do not observe such a difference in performance for the issue type prediction task. Here, seBERT and BERT over flow are very similar to each other. This could mean that the \(F_1\) score of 0.8 is the performance ceiling and a better result is not possible without additional data, e.g., more training data or additional information about the issues. Alternatively, seBERT is penalized by the short sequence length of 128, which gives BERT over flow an advantage for longer issues. This could compensate the difference in model size and data for pre-training. A subgroup analysis of the predictions for issues with less than or equal to 128 tokens and longer issues did not provide conclusive evidence for this. Instead, we observed an instance of Simpson’s paradox [60], i.e., seBERT outperformed BERT over flow on both subgroups, even though the overall performance is about equal. While we could consider this as an indication that the sequence length was not problematic, we cannot provide a definitive answer to this question without extending seBERT to a sequence length of 512 tokens. However, such an extension is, at this time, from our perspective not warranted due the required energy consumption (see Section 7.3) and we believe that the computational effort for this should only be spent if there is a clear expected advantage of considering longer sequences.

The difference between the \(\text{BERT}_{\text{BASE}}\) and \(\text{BERT}_{\text{LARGE}}\) model further shows that general-domain models – especially large general domain models – should be used with caution: often, this model converged to a trivial result where all instances were classified as negative. The logs of the training showed that this happened when the fine-tuned model was still very bad after the first epoch. In the subsequent epochs, the optimization found that a trivial model was actually better in terms of accuracy and cross-entropy than the first attempts. Unfortunately, we cannot determine the exact reason for this. However, we believe that this may be due to the combination of a small data set for fine-tuning and a lack of SE domain understanding of the general model. As a result, the training did not converge towards a reasonable solution within one epoch. Subsequently, the

---

\[\text{Fig. 9. Results of sentiment mining on the GH data.}\]
optimizer got stuck in the local optimum of the trivial model.\footnote{We cannot rule out that other training parameters (e.g., batch size, learning rate), would yield better results with BERT\textsubscript{LARGE} or any other of our models. However, this does not affect our conclusions, as we discuss in the threats to validity (see Section 7.5).} That this did not happen with the equally large seBERT provides another indication that the SE specific pre-training, in fact, the opposite happened: since the pre-training already captured the domain very well, seBERT usually achieved the optimal result within one, at most two epochs, regardless of the model size.

Together with the results from Tabassum et al.\cite{Tabassum:CoRR:2019} for the NER task that showed that the BERT\textsubscript{overflow} model was also better than a BERT\textsubscript{BASE} model, we have a clear answer for RQ2.

**Answer to RQ2:** Transformer models pre-trained with SE domain data consistently outperform other models on SE use cases and should be considered as the state-of-the-art. Pre-trained models from the general domain should be used with care, especially if only a small amount of data is available for the fine-tuning. However, we also found that SE domain models perform similar to general domain models for sentiment mining on SE data, i.e., a use case that does not require SE specific knowledge.

### 7.3 Ethical Considerations

Large deep learning models for NLP, like the transformer models we consider within this work, are associated with several ethical challenges, as is, e.g., highlighted in the famous stochastic parrots paper by Bender et al.\cite{Bender:2018}. Due to impact of Bender et al.\cite{Bender:2018} and the direct relation to BERT models, we structure our consideration of ethical aspects following the four major ethical concerns highlighted in their work.

The first aspect highlighted by Bender et al.\cite{Bender:2018} is that, in the nature of our work, humans learn continuously about new tools, i.e., there is no guarantee that the SE specific pre-training that was optimized for the hardware we were using and by restricting the sequence length to 128 tokens. Based on the energy consumption of the system we used for training that requires between 2.5 kW and 3.5 kW when utilized fully, we estimate that we required between 270 kW and 378 kWh for the 4.5 days of pre-training. Under load the cooling of the data center, where the compute nodes are located, has a Power Usage Effectiveness (PUE) of about 1.23. Additional overhead regarding storage and network has also to be taken into account, so we are calculating with an overhead of approximately 30\%. Hence, we estimate that we consumed at most 378 kWh - 1.30 = 491.4 kWh. Based on the 366 g CO\textsubscript{2} that is generated per kWh in 2020 in Germany\cite{Fernández-García:2020}, this means the pre-training produced up to 180 kg of CO\textsubscript{2}. This is roughly the amount of the CO\textsubscript{2} for one tank filling of an average family car, which we believe is not unreasonable from an ecological perspective for a one-time effort. In comparison, Strubell et al.\cite{Strubell:2019} estimate that they required about 1507 kWh for the training of a BERT\textsubscript{BASE} model. This means that we could train the seBERT model with only one third of the environmental impact than a normal BERT\textsubscript{BASE} model, even though we use a BERT\textsubscript{LARGE} architecture and 119.7 Gigabyte instead of 16 Gigabyte of textual data for training.

The second aspect highlighted by Bender et al.\cite{Bender:2018} is the quality of the training data. The authors highlight that the size of the training data does not guarantee diversity, data collected from the past can, by definition, capture how language evolves, and that there is a risk that models, therefore, capture and enforce existing biases. We did not systematically evaluate BERT\textsubscript{overflow} and seBERT for such biases. One aspect we found regardless is that “women” was not in the dictionary of seBERT. Thus, we can be quite sure, even without an in depth consideration that there is at least a severe gender bias within the data, and, consequently, within seBERT. Since BERT\textsubscript{overflow} was trained on less, but similar data, we believe that there should be a similar gender bias. We cannot comment on other biases, e.g., racial bias or similar. We also note that we cannot exclude that men is only included in the seBERT vocabulary because it is polysemous (man page). Tasks like NER or classification of bugs should not be affected by such biases and can safely be used. Tasks where bias is be relevant, e.g., sentiment mining of developers, the development of chat bots or automated answering of SE questions should only be conducted after a detailed consideration of such ethical considerations. For tasks where the impact of bias is unclear, e.g., the summarizing existing texts within the SE domain, could possibly be used but we still recommend to at least conduct a basic check for such biases.

The third aspect that Bender et al.\cite{Bender:2018} consider is that time may be better spent than on the exploration of ever larger language models. When we transfer this to our work, this means that SE researchers should not invest too much effort into the development of NLP models for the SE domain, but rather focus on the SE data and use cases. For us, this means that we should only provide SE domain models, when machine-learning driven NLP research has major advances, instead of, e.g., trying to directly find new transformer architectures for SE benchmarks with SE data. Currently, it seems like the SE community is already using such an approach, as there are only few domain specific models and they are all re-using architectures that were developed by researchers for the general domain (see Section 3).

The fourth issue raised by Bender et al.\cite{Bender:2018} is that, in the end, such large NLP models are nothing more than stochastic parrots, i.e., models that repeat what they have seen in the data, with some random component. This is due to our lack of understanding about the internal structure and reasoning within models with millions of parameters. This is also a property of SE domain models that should be respected for any later use of these models in an ethical way. For example, we found that the SE domain models are very good at suggesting technologies. When we use the sentence “You can use [MASK] for code coverage in Java,” seBERT and BERT\textsubscript{overflow} predict tools like Cobertura, gcov, Emma, or Jacoco. We found that this works with different languages and different kinds of tools. However, using the models within a chat bot or Q&A system to recommend suitable tools means that the past is encoded and developers of new tools would not have a chance, unless the model is retrained. In comparison, humans learn continuously about new tools, i.e., there
would not be such an ethical problem. The consequence of this issue is similar to the impact of potential biases: we recommend to be careful when using the NLP models to generate responses to actual queries, unless it was determined that the possible responses are carefully validated for the given use case.

7.4 Open Issues
Our work demonstrates that NLP models pre-trained with SE domain data are useful. However, there are also limitations to the understanding of transformers that we established. Since we used BERT both as general domain reference model, as well as the architecture for our models, it is unclear how the results generalize to other transformer models. While the difference to models that have a similar size like RoBERTa [18] should be relatively small, it is unclear if extremely large models like GPT-3 [19] may be able to correctly understand the meaning of texts both in the SE and general domain, same as human experts. While there is no reason that this should be the case, there is also no strong argument against this. However, currently the scale of these models is beyond almost anyone, except the largest labs and companies of the globe [64]. Therefore, even if this were the case, such improvements could currently only be harnessed by a small elite. Thus, we believe that for the majority of SE researchers and vendors who may consider building NLP capabilities into their tools, models like BERT are a more realistic, current alternative.

A related limitation is the impact of the context length on the results. Within this work, we work with a maximum context length of 512 tokens (BERT, BERTOverFlow) and 128 tokens (seBERT). While we argue that most SE texts in our data are shorter anyways, this is also due to the type of text we consider. If we were to, e.g., consider README files instead, we would likely have many examples of longer texts. Thus, while our results show that even the relatively short context of 128 tokens is sufficient and actually leads to the best results in both domain-dependent fine-tuning examples we consider, this may not generalize to NLP tasks on longer inputs. Especially generative tasks, like summarizing long documents, may benefit from a longer context that is sufficient to capture the meaning of the whole text at once. In case studies find that models with shorter contexts, like BERT, do not generate suitable results for longer documents, other transformer models, like Big Bird [65], which achieves a sequence length of 4096 tokens, could be used.

Another limitation is a corollary from our statement regarding longer documents: while our corpus is already relatively diverse, with Stack Overflow, issues, and commit messages, this still does not capture the whole SE domain and, most notably, lacks examples with longer texts. Thus, even if we wanted to train a model with a longer sequence length, this could only make a difference on a small fraction of this type of SE data and the available data would likely not be sufficient to correctly model longer contextual relationships. Unfortunately, there is neither a suitable data set that could be exploited for pre-training, nor is there a benchmark task for NLP within the SE domain that requires longer texts. Thus, to advance NLP for the SE domain for tasks with long sequences, our community would first need to solve the associated data challenge, both with respect to data for pre-training, as well as through a curated data set that is suitable for the benchmarking of a fine-tuned application. These limitations are not only restricted to length, but also to tasks that actively exploit the capability of BERT models to compare two sentences by exploiting the semantics of the [SEP] token. Thus, further fine-tuning tasks that evaluate the utility of BERT for such problems, e.g., the identification of related Stack Overflow posts [66], still need to be considered. Additionally, similar fine-tuning tasks to the ones we considered to evaluate the generalization of the results to similar tasks, e.g., app review classification [67], which is similar to sentiment mining.

Finally, we already highlighted the lack of an evaluation of the SE models from an ethics perspective. While this was not within the scope of our work, future work must deal with potential biases in SE domain models, unless their usage is restricted to few and possibly uncritical use cases, as is, e.g., the case in our fine-tuned examples. We note that this does not only affect seBERT and BERTOverFlow, but also models like CodeBERT [68], in case this is used to generate texts, e.g., automated documentation generation. From our perspective, this is a precursor of any type of generative NLP application, i.e., NLP models that actively generate texts, e.g., to answer questions like “what does this code do” or “summarize this README file,” but also for any application like sentiment mining, which is known to encode biases [69].

7.5 Threats to Validity
We report the threats to the validity of our work following the classification by [70] suggested for software engineering by [71]. Additionally, we discuss the reliability as suggested by [72].

7.5.1 Construct Validity
The construct of our validation of NLP models may be unsuitable. The direct comparison of WordPiece vocabularies neglects to account for the internal structure of the models. For example, the internal structure could achieve that the combination of tokens “wo##men” is the same as having the token “women” directly in the vocabulary. We address this issue by not only considering the overlap and tokenization, but also through a qualitative analysis of the non-overlapping words. Our data indicates that these are mostly domain specific terms, which makes sense given the training data and also means that an effect where these words are known by the model, regardless of them missing in the vocabulary. Similarly, our study of the contextual interpretation of sentences and words through the MLM task may be unsuitable. Our lack of consideration of different parameters for training (batch sizes, learning rates, etc.) may lead to sub-optimal models after fine-tuning, which could affect our conclusions.

7.5.2 Internal Validity
While we have good reason to believe that the differences between the models we observe are due to the difference in the data used for pre-training the models, we cannot rule out that there may be other reasons for these differences, due to the black box nature of the models. Most threats to
7.5.4 External Validity

Since our consideration of fine-tuning only considers five classification tasks, we cannot with certainty conclude that the SE domain models are also better than the general domain models for other SE tasks, e.g., summarizing content or comparing sequences. However, since our results indicate that SE-specific models are better at capturing SE specific aspects and because prior work also found a similar result regarding the NER task [10], we believe that this is unlikely. Moreover, as already discussed in Section 7.4, our results may not generalize to extremely large models, as they may be able to capture multiple domains correctly, due to their size. We note that our pre-training and evaluations are limited to data collected from open source projects. While many open source projects are developed professionally, there may still be issues due to unknown in-house terminology when these models are used within a proprietary context that requires such knowledge. Our results do not allow for conclusions regarding the question if fine-tuning of open source models would be sufficient for such use cases.

7.5.5 Reliability

The definition of the positive, neutral, and negative sentences for the assessment of the contextual embeddings may influence our results, i.e., other researchers would almost certainly not have selected exactly the same sentences as us and, therefore, may have a different view on the capability of the models with respect to understanding the context of statements. However, while other sentences could always lead to differences, we note that we did not conduct any cherry picking to show the most differences, but rather first defined the sentences and then applied the models. Moreover, our playground provides anyone with the ability to evaluate the differences in the predictions of masked words between the models. Additionally, we augmented the analysis on the curated corpus with a pure empirical analysis, in which we randomly sampled sentences with words with special meaning in the SE domain and then evaluated the ability of the models to predict the missing words. While this does not give us insights into the inner workings of the models, these empirical results provide further evidence that our results are not biased by our selection.

8 Conclusion

Within our work, we find that NLP for SE can benefit from transformer models pre-trained with SE data. While this was already known for applications that include source code, there was only little evidence for other natural language tasks. Through our work we not only explored the differences in the expected performance of applications, but also the tried to understand if the models really have a better understanding of the SE domain. For this, we manually compared the vocabularies of general domain and SE domain BERT models and compared how these models completed sentences with masked words. This analysis showed that while general domain models have a rough

18. The Jira Issues only contain data until 2014 and the difference in performance of predictions is only later data is considered is small [11]

19. https://smartshark2.informatik.uni-goettingen.de/sebert/index.html
but imprecise understanding of the SE domain, the SE models are more precise. This improved understanding also led to significantly better results in fine-tuned within the SE domain, but not in a general task like sentiment mining applied to SE data. In conclusion, we recommend to ensure that a large amount of SE data was used for pre-training large NLP models, when these models are used for SE tasks.

ACKNOWLEDGMENTS

The authors would like to thank the GWDG for their support regarding the usage of the GPU resources required for this article.

REFERENCES

[1] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 6000–6010.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19–1423
[3] Z. Lan et al., “Pretrained Language Understanding,” Red Hook, NY, USA: Curran Associates Inc., 2019.
[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” 2019. [Online]. Available: https://paperswithcode.com/paper/langua-

ge-models-are-unsupervised-multitask
[5] S. Nagel, Co-news, 2016. [Online]. Available: https://commoncrawl.org/2020.acl-main.703
[6] I. Beltagy, K. Lo, and A. Cohan, “SciBERT: Pretrained language model for scientific text,” in Proc. Conf. Empir. Methods Natural Lang. Process., 2019, pp. 3613–3618.
[7] J. Lee et al., “BioBERT: A pre-trained biomedical language representation model for biomedical text mining,” Bioinformatics, vol. 36, no. 4, pp. 1234–1240, 09 2019.
[8] K. Huang, J. Allosaar, and R. Ranganath, “ClinicalBERT: Modeling clinical notes and predicting hospital readmission,” 2019, arXiv:1904.05342.
[9] V. Efstathiou, C. Chatzilenas, and D. Spinellis, “Word embeddings for the software engineering domain,” in Proc. 15th Int. Conf. Mining Softw. Repositories, 2018, pp. 38–41.
[10] J. Tabassum, M. Maddela, W. Xu, and A. Ritter, “Code and named entity recognition in StackOverflow,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics: Syst. Demonstrations, 2019, pp. 37–42.
[11] S. Herbold, A. Trautsch, and F. Trautsch, “On the feasibility of automated prediction of bug and non-bug issues,” Empir. Softw. Eng., vol. 25, no. 6, pp. 5333–5369, Sep. 2020.
[12] K. Herzig, S. Just, and A. Zeller, “It’s not a bug, it’s a feature: How misclassification impacts bug prediction,” in Proc. Int. Conf. Softf. Eng. Test. Meas., 2013, pp. 392–401. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486840
[13] A. Trautsch, J. Erbel, S. Herbold, and J. Grabowski, “A commitment-level study of static metric value and static analysis warning changes,” 2021, arXiv:2109.03544.
[14] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, “Sentiment analysis for software engineering: How far can pretrained transformer models go?”, in Proc. IEEE Int. Conf. Softw. Maintenance Evol., 2020, pp. 70–80.
[15] J. Eisenstein, Introduction to Natural Language Processing. Ser. Adaptive Computation and Machine Learning Series, Cambridge, MA, USA: MIT Press, 2019.
[16] M. Lewis et al., “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,”Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703
[17] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite bert for self-supervised learning of language representations,” in Proc. Int. Conf. Learn. Representations, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AElvS
[18] Y. Liu et al., “RoBERTa: A robustly optimized BERT pretraining approach,” 2019, arXiv:abs/1907.11692.
[19] T. B. Brown et al., “Language models are few-shot learners,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 1877–1901.
[20] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” J. Mach. Learn. Res., vol. 21, no. 126, pp. 1–39, 2021.
[21] L. Tunstall, L. von Werra, and T. Wolf, Natural Language Processing With Transformers, Newton, MA, USA: O’Reilly Media, Inc., 2021.
[22] J. Vig, “A multiscale visualization of attention in the transformer model,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics: Syst. Demonstrations, 2019, pp. 37–42.
[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in Proc. 1st Int. Conf. Learn. Representations, 2013.
[24] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, “Sentiment strength detection in short informal text,” J. Amer. Soc. Inf. Sci. Technol., vol. 61, no. 12, pp. 2544–2558, 2010.
[25] M. R. Islam and M. F. Zibran, “Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text,” in Proc. 25th ACM Int. Conf. Softw. Anal. Design Eng., 2018, pp. 125–146, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121218301675
[26] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, “SentiCR: A customized sentiment analysis tool for code review interactions,” in Proc. 32nd IEEE/ACM Int. Conf. Automated Softw. Eng., 2017, pp. 106–111.
[27] C. Cai et al., “Deep, Fast, and Accurate: Sentence-Pair Classification” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18–1202
[28] S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou, “Distributional semantics resources for biomedical text processing,” in Proc. Lang. Biol. Med., 2013, pp. 39–44.
[29] S. Ghosh, P. Chakraborty, E. Cohn, J. S. Brownstein, and N. Ramakrishnan, “Ch-AIN: Online, distributed, unstructured text: A vocabulary driven word2vec approach,” in Proc. 25th ACM Int. Conf. Knowl. Manage., 2016, pp. 1129–1138.
[30] D. Araci, “Finbert: Financial sentiment analysis with pre-trained language models,” 2019, arXiv:1908.10063.
[31] A. Ferrari and A. Esuli, “An NLP approach for cross-domain ambiguity detection and requirements engineering,” Automated Softw. Eng., vol. 22, no. 3, pp. 328–352, Dec. 2015.
[32] B. Theeten, F. Vandeputte, and T. Van Cutsem, “Import2vec: Learning embeddings for software libraries,” in Proc. 16th Int. Conf. Mining Softw. Repositories, 2019, pp. 18–28.
[33] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, “Code2vec: Learning distributed representations of code,” Proc. ACM Program. Lang., vol. 3, no. POPL, pp. 1–29, Jan. 2019.
[34] Z. Feng et al., “Code2vec: A pre-trained model for programming and natural languages,” in Proc. Findings Assoc. Comput. Linguistics, 2020, pp. 1536–1547.
[35] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: Pre-training text encoders as discriminators rather than generators,” in Proc. Int. Conf. Learn. Representations, 2020.
[36] Stack Exchange, Stack exchange data dump, 2014. [Online]. Available: https://archive.org/details/stackexchange
[37] Internet Archive, Internet archive, 2010. [Online]. Available: https://archive.org/index.php
[38] Stack Exchange, Stack overflow public dataset, 2016. [Online]. Available: https://console.cloud.google.com/marketplace/product/stack-exchange/stack-overflow
[39] I. Grigorik, The GitHub archive, 2012. [Online]. Available: https://archive.org/index.php
[40] Google Cloud, “BigQuery - google cloud platform, 2011. [Online]. Available: https://www.google.com/bigquery
[41] M. Ortu, G. Destedianis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, “The JIRA repository dataset: Understanding social aspects of software development,” in Proc. 11th Int. Conf. Predictive Models Data Analytics Softw. Eng., 2015, pp. 1–4.
et al. Climate Change, Dessau-Roßlau, Germany: Experimentation in Software Engineering Proc. 12th ACM/IEEE Int. Symp. Empir. Softw. Eng. Mass., 2018, pp. 1–10.

W. Maalej, Z. Kurtanovic, H. Nabil, and C. Stanik, “On the automatic classification of app reviews,” Requirements Eng., vol. 21, no. 3, pp. 311–331, May 2016.

Z. Feng et al., “CodeBERT: A pre-trained model for programming and natural languages,” in Proc. Findings Assoc. Comput. Linguistics, 2020, pp. 1536–1547. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.139

S. Kiritchenko and S. Mohammad, “Examining gender and race bias in two hundred sentiment analysis systems,” in Proc. 7th Joint Conf. Lexical Comput. Semantics, 2018, pp. 43–53. [Online]. Available: https://aclanthology.org/S18–2005

T. D. Cook, D. T. Campbell, and A. Day, Quasi-Experimentation: Design & Analysis Issues for Field Settings, vol. 351, Boston, MA, USA: Houghton Mifflin, 1979.

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, Experimentation in Software Engineering, Berlin, Germany: Springer, 2012.

P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empir. Softw. Eng., vol. 14, no. 2, pp. 131–164, 2009.

Julian von der Mosel received the BSc degree from the University of Goettingen. He is currently working as a data engineer.

Alexander Trautsch is currently working toward the PhD degree with the University of Goettingen.

Steffen Herbold is a professor for methods and applications of machine learning with the TU Clausthal.