2022 IEEE 30th International Requirements Engineering Conference (RE) | 978-1-6654-7000-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/RE54965.2022.00046

Retraining a BERT Model for Transfer Learning in Requirements Engineering: A Preliminary Study

Muideen Ajagbe and Liping Zhao
Department of Computer Science, University of Manchester
Oxford Road, Manchester, M13 9PL, UK

***Abstract*—In recent years, advanced deep learning language models such as BERT, ELMO, ULMFiT and GPT have demonstrated strong performance on many general natural language processing (NLP) tasks. BERT, in particular, has also achieved promising results on some domain-specific tasks, including the requirements classification task. However, in spite of its great potential, BERT under-performs on domain specific tasks. In this paper, we present BERT4RE, a BERT-based model retrained on requirements texts, aiming to support a wide range of requirements engineering (RE) tasks, including classifying requirements, detecting language issues, identifying key domain concepts, and establishing requirements traceability links. We demonstrate the transferability of BERT4RE, by fine-tuning it for the task of identifying key domain concepts. Our preliminary study shows that BERT4RE achieved better results than the $\text{BERT}_{\text{base}}$ model on the demonstrated RE task.**

***Index Terms*—Requirements Engineering, Requirements Classification, Language Models, BERT, Domain-Specific Language Models, Transfer Learning, Deep Learning, Machine Learning, Natural Language Processing.**

1. Introduction

Recent developments in transformer-based language models (LMs) [1] such as BERT [2], ELMO [3], GPT [4], ULMFiT [5] represent a major breakthrough in natural language processing (NLP). Replacing static vector representations of words (e.g., word2vec [6]) with contextualized word representations or contextual embeddings, these models have led to significant improvements on almost every NLP task [7], achieving state-of-the-art results in many of the most common NLP benchmark tasks [2], [8]. BERT, in particular, outperforms other LMs [7].

In practical terms, these advanced LMs have realized the long-held dream of transfer learning [9], [10], as they can be pre-trained using *unsupervised learning* on a large quantity of *unlabelled*, *generic* textual corpora, and then *fine-tuned* to perform the downstream tasks, through *supervised learning* on *labelled, task-specific texts*. The transfer learning capability of LMs is very attractive to researchers seeking to use these models to process text intensive tasks such as those in requirements engineering (RE) [11], as these models can be (re)used for tasks other than the task they were trained on, with little fine-tuning effort [12].

However, in spite of the strong performance achieved by these LMs, researchers found that generic LMs do not perform particularly well on domain-specific tasks, as they cannot recognize highly domain-specific vocabulary [13]–[16]. To address this shortfall, several attempts have been made to *retrain* the generic BERT model on domain-specific texts and these attempts have led to improved performance on domain-specific tasks [13]–[16].

In this paper, we present **BERT4RE**, a **retrained** domain-specific LM[1] aiming to support a wide range of requirements engineering (RE) tasks, such as requirements classification, detection of language issues, identification of domain concepts, and establishment of requirements traceability links. BERT4RE is retrained on the generic $\text{BERT}_{\text{base}}$ model [2] using publicly available RE related texts. To demonstrate the transferability of BERT4RE, in this preliminary study, we fine-tune BERT4RE for a specific RE task, that of identifying key domain concepts from requirements text (hence called *concept extraction*). We then compare the performance of BERT4RE with that of $\text{BERT}_{\text{base}}$ by fine-tuning both models on the same labelled dataset.

In this paper, we lay the foundation for building RE specific LMs with the following contributions:

- We propose BERT4RE, a domain-specific LM for RE, and we describe how we retrain this model from the $\text{BERT}_{\text{base}}$ model using RE related data.
- We fine-tune both BERT4RE and $\text{BERT}_{\text{base}}$ by using a labelled dataset and then apply them to a multiclass classification task of identifying nine requirements concepts.
- We show that BERT4RE outperforms $\text{BERT}_{\text{base}}$ and discuss the results obtained by each of these models. We make BERT4RE available on Zenodo [18] for RE researchers and practitioners to experiment with.

The paper is structured as follows: Section 2 discusses related work on retraining and fine-tuning BERT models for domain-specific tasks. Section 3 describes how we retrain BERT4RE using RE related datasets. Section 4 presents an experimental study in which we fine-tune both BERT4RE and $\text{BERT}_{\text{base}}$ on a common dataset to compare their performance. Section 5 reports our study results, whereas Section 6 discusses potential validity threats to our research. Finally, Section 7 concludes the paper and outlines how we plan to carry forward this research.

[1]We distinguish between a pretraining and retraining, with the former referring to pretraining a LM on generic data and the latter meaning retraining a pretrained model on domain-specific data. The idea of model retraining is similar to domain adaptation [17].

978-1-6654-7000-1/22/$31.00 ©2022 IEEE
DOI 10.1109/RE54965.2022.00046

2. Related Work

In this section, we briefly recount some of closely related work, focusing on the work on retraining and fine-tuning BERT models for domain-specific tasks.

A. Retraining BERT Models

As already stated, the BERT model has been retrained for several domains to improve its performance on domain-specific tasks. In this section, we briefly summarize several efforts to retrain domain-specific BERT models.

To our knowledge, Sainani et al. [13] reported the only effort in RE to retrain BERT. The authors retrained the $\text{BERT}_{\text{base}}$ model to classify the contractual obligations into different requirement types. They used the same software contracts dataset to retrain and fine-tune the BERT model. They found that the rertained BERT outperformed other learning techniques (SVM, Random Forest, Naive Baye, and BiLSTM) by a small margin. However, as the authors used the same dataset for model retraining and fine-tuning, the retrained model is task-specific, not domain-specific, as the model has only been retrained on a task-specific dataset.

Outside RE, there are more efforts to retrain the BERT model. These include BioBERT [14], where the $\text{BERT}_{\text{base}}$ model was retrained on a corpus of biomedical articles from pubMEd for 470,000 iterations. BioBERT has shown to improve the performance on classifying biomedical texts than the generic BERT model.

The CLINICALBERT [19] and CLINICALBioBERT [20] are retrained on clinical text from MIMIC-III dataset [21] and fine-tuned on BioBERT respectively for 150k steps. These models outperformed the $\text{BERT}_{\text{base}}$ model.

Also, SCIBERT [15] is another BERT model variant retrained on scientific text from the semantic scholar corpus. Specifically, the model was retrained on 1.14M biomedical and computer science texts. SCIBERT was developed by retraining $\text{BERT}_{\text{base}}$ model using its original vocabulary and another variant, SCIVOCAB was developed using an in-domain vocabulary by retraining BERT from scratch.

BERTweet [22] is a variant of $\text{BERT}_{\text{base}}$ model retrained on English tweets. BERTweet was retrained for 950,000 iterations. The model has shown to improve the performance on POS and NER task than other LM such as RoBERTa and XLM-R.

Another BERT variant model is the LEGAL-BERT [16] which was developed by running addition retraining steps on $\text{BERT}_{\text{base}}$ with English legal text. LEGAL-BERT was fine-tuned on annotated legal dataset for text classification and sequence tagging task with impressive results.

B. Fine-Tuning BERT Models

Most published works in RE focus on using fine-tuned BERT models to perform RE tasks. One of these works is by Hey et al. [12], who propose NoRBERT, a transfer learning approach for requirements classification. NoRBERT applies fine-tuned BERT models ($\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$) to both binary and multiclass classification tasks. Experiments show that NoRBERT can achieve an F1-score of over 90% on most frequent non-functional requirements (NFR) classes such as Security and Performance; NoRBERT generally outperforms traditional, non-transfer learning approaches.

Li et al. [23] propose DEMAR, a deep multi-task learning approach for requirements analysis. DEMAR fine-tuned the $\text{BERT}_{\text{base}}$ model to identify and annotate requirements from open forum. Experiments show that on eight open source projects, DEMAR achieved a Precision of 91% and Recall of 83% on requirements discovery task to outperform traditional machine learning approaches.

Also, Wang et al. [24] propose DEEPCOREF, a transfer learning approach for coreference detection in requirements. DEEPCOREF fine-tuned $\text{BERT}_{\text{base}}$ model to detect and resolve entity coreference in requirements. Experiments on requirements documents show that DEEPCOREF can achieve a Precision and recall of over 96% respectively on industry partner data. DEEPCOREF outperforms traditional and non-transfer learning approaches.

RE-BERT by Araujo and Marcacini [25] fine-tuned the $\text{BERT}_{\text{base}}$ model to extract semantic representation of requirements from app reviews. Experiments on manually labelled data of eight different apps show that RE-BERT outperforms existing tradition and non-transfer learning SOTA results with F-score of 62%.

Similarly, Pota el al. [26] propose the fine-tuning of $\text{BERT}_{\text{base}}$ model to support sentiment analysis on Italian tweets. Experiments show that the proposed approach outperforms other transfer learning and non-transfer learning approach on an annotated Italian sentiment data with F-score of 75%.

3. BERT4RE: Retraining BERT for RE

A. Retraining Method

The generic version of the BERT LM was pre-trained on Wikipedia (2,500 million words) and BooksCorpus (800 million words) [2]. The pre-training involved getting the BERT model to learn two unsupervised tasks known as "Masking LM (MLM)" and "Next Sentence Prediction (NSP)". BERT was pretrained in two basic model sizes: $\text{BERT}_{\text{base}}$ (L=12, H=768, A=12, Total Parameters=110M) and $\text{BERT}_{\text{large}}$ (L=24, H=1024, A=16, Total Parameters=340M), where "L" is the number of Transformer blocks or layers, "H" is the hidden size, and "A" is the number of self-attention heads.

In our current preliminary research, we follow the previous work involving retraining BERT (see Section 2.1) and chose $\text{BERT}_{\text{base}}$ as our model size. We retrain this model on RE related datasets through unsupervised learning, for the model to learn the domain-specific vocabulary given by these datasets. This retraining process, as shown in Figure 1, is basically the same as the one used to train the original $\text{BERT}_{\text{base}}$ model. We retrained all 12 attention layers, one by one. The retraining also entails the model to learn the MLM and NSP tasks on the domain-specific vocabulary. The outcome of the retraining process is a RE-specific $\text{BERT}_{\text{base}}$ model called BERT4RE.

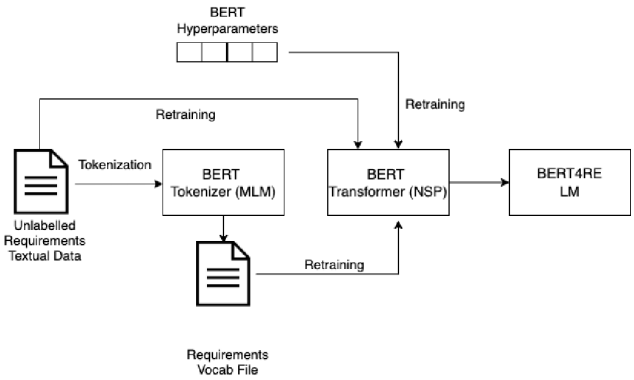

Fig. 1. The unsupervised learning process for retraining BERT for RE. The outcome of the retraining is a RE specific model called BERT4RE.

The subsections below describe what data are used in our retraining and how we prepare them.

B. Retraining Data

1) Data Collection: To retrain the BERT model, we collected the following four requirements related datasets from different sources:

- PROMISE NFR Dataset [27]: This is a popular dataset widely used by RE researchers for training and testing their machine learning classifiers [12], [28]–[30]. The dataset contains 625 requirements (255 FRs and 370 NFRs) from 15 software development projects.
- PURE Dataset [31]: This is one of the largest RE datasets. The dataset contains 522,444 lexical words and 865,551 tokens. We only collected part of this dataset related to FRs, as these requirements statements are well structured and easy to identify. The extracted requirements contains 29,000 unique words.
- App Review Datasets: We downloaded four million app reviews from the Appfollow website (appfollow.io). These reviews were submitted to from January 2010 to April 2021. We extracted over three million unique words from these reviews.
- Google Playstore App Reviews[2]: We collected around 600 thousand app reviews from this app store and extracted more than two million unique words from these reviews.

These datasets, summarized in Table I, contain a total of 7,758,173 (more than **7 millions**) words. We combined all these datasets into one single dataset (corpus) for retraining $\text{BERT}_{\text{base}}$.

2) Data Pre-processing: The retraining dataset needs to be cleaned and processed into word tokens and sentences, before they can be used for retraining the BERT model. To do so we applied the following NLP pipeline to the texts in the dataset:

- Tokenization: This step converts text into tokens. We used the StanfordCore NLP tokenizer[3] to break our input requirements text into word tokens. We conceded that app reviews are closer to Tweets in verbiage, hence we use NLTK TweetTokenizer toolkit for the tokenization of our app requirements related app review documents due to lack of tokenizers specifically tailored to them.
- Emoji/Emoticon Conversion: The emojis and emoticons are used in the app reviews to convey and represent the mood of the reviewers. This means that emoticons are strong expression of reviews which needs to be translated into text. To process out emoticons in app reviews, we use the python emoji package [4] to translate emoticons into text string. Here, every emoji icons is translated into a word token.
- Normalization: This process involves the transformation of text or icons into a natural language form thereby reducing its randomness. To achieve this, we filtered the app reviews documents using the langdetect python library[5] from Google's language-detection library to detect only English language reviews. The second step of our normalization includes the conversion of our text to lowercase, the removal of numbers and special characters, punctuation and white-spaces. Sentences that are less than 2 words were removed and URL links were converted into special tokens.
- Sentence Splitting: This step splits text into sentences based on the period delimiter by using the StanfordNLP Sentence Splitter[6].

The output of this pipeline consists of a set of pre-processed requirements and requirements related texts. These textual data can now be used for retraining the BERT model.

TABLE I
REQUIREMENTS DOCUMENTS FOR RETRAINING BERT4RE.

Dataset	#Words	Description
PROMISE NFR Dataset	10,629	652 labelled functional and non-functional requirements
PURE Dataset	38,420	79 requirements documents in several application domains such as health, rail, etc.
App Reviews	5,101,685	A collection of large cache of app reviews on social media application such as twitter, whatsapp and snapchat.
Playstore Google Apps User Review	2,607,439	A dataset of user reviews for applications on Google play store apps.

C. Retraining Procedures

To retrain BERT4RE model, we perform the following steps:

First, to generate each training sentence, we automatically scan through our dataset to determine the maximum length (i.e., the maximum number of words) of the sentences. We

[2]https://www.kaggle.com/gauthamp10/google-playstore-apps
[3]https://stanfordnlp.github.io/CoreNLP/tokenize.html
[4]https://pypi.org/project/emoji/
[5]https://pypi.org/project/langdetect/
[6]https://stanfordnlp.github.io/CoreNLP/ssplit.html

find that the longest sentence contains 90 words. Following Nguyen et al. [22], we used FastBPE [32] to segment each sentence in our dataset using a generated vocabulary of 32,000 subwords. Each subword is a part of the word that conveys some meaning. For example, the word ”subword” has two subwords: ”sub” and ”word”. The average number of subword tokens per sentence in our dataset is 25. The reason for splitting words into subwords is to make words finer grained to make it easier for tokenization and to give the LM more unknown or rare words.

Next, we followed RoBERTa [33] and retrain the $\text{BERT}_{\text{base}}$ and define hyperparameters for retraining. We set a batch size of 128 sequences and 3200 tokens/batch (128 * 25 subword tokens). For the iteration steps, we multiply the number of words in our datasets (7 millions) by the number of subword tokens (25) and divide it by our batch size (128) to generate our sequence blocks (7M * 25/ 128 = 1.3M sequence blocks). We retrain the $\text{BERT}_{\text{base}}$ for 40 epochs equivalent to 1.3M * 40/ 3200 = 17, 000 steps.

Lastly, we use Adam [34] for optimization with a learning rate of 1e-4, a weight decay of 0.01 and a dropout rate of 0.1. The retraining was performed by using Google Cloud TPU v3-8 with a single 8-core processor.

4. Fine-Tuning BERT4RE and $\text{BERT}_{\text{base}}$

A. Study Method

This section presents an experimental study in which we fine-tune both BERT4RE and $\text{BERT}_{\text{base}}$ to perform a multiclass classification task of identifying key domain concepts from requirements text. The aim of this study is to demonstrate the transferability of BERT4RE and compare the performance of a retrained model with that of a pretrained model. Below, we describe the dataset used for fine-tuning these two models, the fine-tuning procedure, and the evaluation methods.

B. Fine-Tuning Dataset

The dataset used for fine-tuning is based on the PURE dataset [31]. We collected a total of **1,055 FRs sentences** from this dataset. The extracted FRs contain **22,916 words**, with a maximum length of sentences being 90 words. The extracted FRs are lemmatized, POS tagged and parsed to enable easy labelling. We then manually label each requirement according to nine semantic concepts (or cases) extracted from Fillmore's Case Frame [35] and the SOM patterns [36]. In what follows, we briefly introduce these nine concepts.

- **Agent (AGT)**: This is the inanimate initiator or the subject of an action that causes changes to an object. For example, in ”the system shall save the users info to the database”, the word ”system” is an agent.
- **Object (OBJ)**: This is the key object for which an action is performed. For example, in ”the system shall save the user contact details to the database with specified roles”, the term ”user contact details” is the key object.
- **Instrument (INT)**: This is the tool used by the agent in order to carry out an action. For example, in ”the vision-impaired user should be able to read the text with a special magnifier function”, the term ”special magnifier function” is the instrument.
- **Container (CONT)**: This is the structural object that stores information used by an agent. For example, in ”the system shall ensure that user input is automatically stored in a file”, the term ”file” is the container.
- **Dative (DAT)**: The dative case is the case that shows the indirect object of an action. For example, in ”the system shall send the user an email confirmation after receiving the payment from the user,” the term ”email confirmation” is the dative case.
- **Location (LOC)**: This is the place identified by an action. For example, in ”the project team shall meet every Tuesdays between 9am and 10am in the staff room”, the term ”staff room” is the location.
- **Temporal (TEMP)**: This is the occurrence of time, date or frequency expressed through an action. For example, in ”the project team shall meet every Tuesdays between 9am and 10am in the staff room”, the term ”every Tuesdays between 9am and 10am” is temporal.
- **Goal (GOAL)**: This is the intended effect to be achieved through an action. For example, in ”the system shall assign each registered user with a unique tag”, the term ”unique tag” is the goal.
- **Constraint (CONST)**: This is the condition on an action. For example, in ”the system shall notify the user if their personal information has been changed”, the term ”if their personal information has been changed” is the constraint.

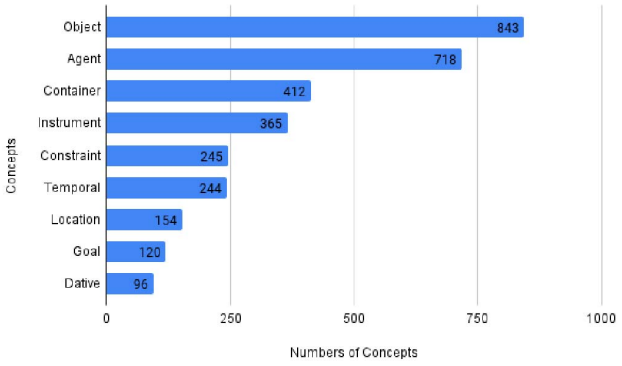

Fig. 2. The instances of individual key domain concepts used in labelling our fine-tuning dataset.

Figure 2 shows the distributions of the these concepts in the extracted FRS and their instances.

C. Fine-Tuning Procedures and Evaluation Methods

To fine-tune both BERT4RE and $\text{BERT}_{\text{base}}$ models to perform a **multiclass classification task with 9 classes** (i.e., concepts), we take the following steps:

First, we prepare the labelled dataset using Python's module Pytorch Dataset Class to represent the sentences in the dataset into a 1055×90 array (**1055 requirements sentences** and a **maximum of 90 words per sentence**).

Next, we define the hyperparameters of each models for fine-tuning based on the original $BERT_{base}$ [2]. Specifically, we set the batch size for the dataset as 16, the maximum length of sentence as 90, the number of labels as 9, the learning rate as 1e-4, the epochs as 20, the weight decay as 0.01 and the dropout rate as 0.1. We also define the tokenizer of each model as BertTokenizerFast.from_pretrained('bert-base-uncased').

Afterwards, we split the array randomly into a training and testing set, with 70-30 splits. Finally, we fine-tune each model using the training set and then test each fine-tuned model using the testing set. Both fine-tuning and testing involve getting these models to classify each input sentence into one or more categories according to the nine labels. The fine-tuning of each model was done using the defined hyperparameters.

After fine-tuning both models, we assess its training loss through a metric used to determine how a model fits the fine-tuning dataset. The smaller training loss means the better model fitting. We found that BERT4RE has an average training loss of 0.0007, which is significantly lower than the training loss of $BERT_{base}$, which is an average of 0.005.

To measure the performance of BERT4RE and $BERT_{base}$ on individual classes (i.e., concepts), we use the unweighted precision (P), recall (R) and F1-score metrics to measure the testing results of these models. More specifically, for precision, we measure the percentage of the correctly classified occurrences of concept in relation to the total numbers of concept identified. For recall, we measure the percentage of the correctly classified occurrences of concept. For the F1-Score, we measure the performance of the fine-tuned model by aggregating precision and recall measures through a harmonic mean [37].

To measure the overall performance of each model on all the classes as a whole, we use both *macro-averages* and *micro-averages* [37]. Specifically, the macro-average computes precision, recall and F1 independently for each class and then takes the average of each metric, thus treating all the classes equally regardless of their size. The macro-averages thus reduce the effect of the imbalanced classes on the performance results. On the other hand, the micro-average aggregates the contributions of all classes to compute the average precision, recall and F1. Consequently, the micro-averaged precision, recall and F1 are both equal to the *accuracy*.

Table II presents the performance results achieved by the fine-tuned BERT4RE and $BERT_{base}$ models on individual classes as well as their macro and micro-averages. The table is ordered by the F1-scores of BERT4RE. In the next section, we discuss these results.

TABLE II
PERFORMANCE RESULTS ACHIEVED BY FINE-TUNING BOTH $BERT_{BASE}$ AND BERT4RE ON IDENTIFYING 9 REQUIREMENTS CONCEPTS.

	$BERT_{base}$			**BERT4RE**		
Concept	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
AGT	0.94	0.88	0.91	0.96	0.94	0.95
CONT	0.88	0.94	0.91	0.92	0.94	0.93
INT	0.86	0.89	0.88	0.92	0.89	0.91
TEMP	0.74	0.95	0.83	0.87	0.95	0.91
OBJ	0.85	0.91	0.88	0.90	0.92	0.91
LOC	0.80	0.92	0.85	0.86	0.91	0.88
GOAL	0.75	0.73	0.74	0.93	0.79	0.86
CONST	0.89	0.89	0.89	0.90	0.82	0.86
DAT	0.88	0.45	0.59	0.89	0.81	0.85
Accuracy			0.88			0.95
Macro Avg	0.84	0.86	0.85	0.92	0.91	0.92
Micro Avg	0.88	0.88	0.88	0.95	0.95	0.95

5. RESULTS

We can make the following observations from Table II:

- BERT4RE's macro-averaged P, R, and F1 are 7% higher than those of $BERT_{base}$, suggesting BERT4RE handles the imbalanced classes in the fine-tuning dataset better than $BERT_{base}$.
- BERT4RE's micro-averaged P, R, and F1 are 7% higher than those of $BERT_{base}$, suggesting BERT4RE outperforms $BERT_{base}$ and is more accurate.

When looking at the performance results of these two mod8ls on the individual classes, Table II also reveals the following findings:

- BERT4RE outperforms $BERT_{base}$ on 8 out 9 classes (Agent, Container, Instrument, Dative, Location, Object, Goal, and Temporal) and achieves over 90% F1 scores on the five classes (Agent, Container, Instrument, Object, and Temporal). As Location, Goal and Dative are smaller classes, the performance of BERT4RE over $BERT_{base}$ on these classes shows that BERT4RE not only outperforms the state-of-the-art model but also improves over the performance on the imbalanced classes. In particular, BERT4RE outperforms $BERT_{base}$ on the smallest class Dative, with a F1-score of 85% – 26% more than that of $BERT_{base}$.
- BERT4RE achieves 3% lower F1 score than $BERT_{base}$ on the Constraint class. This is perhaps due to that Constraint class usually has the sentence structures such as "If ..., then ..." or "While ..." and these structures have already been learned by the original, pretrained BERT model. By contrast, most sentences in our retraining dataset are simple sentences without the words "If", "then" and "While", so the retrained model has a limited vocabulary than the original, pretrained BERT model.

The above results clearly demonstrate the benefit and potential of retraining the $BERT_{base}$ model, even with a small amount of data.

6. THREATS TO VALIDITY

This section discusses the potential validity threats to our work. These threats concern our retraining dataset and fine-tuning dataset, our experimental design of model retraining and fine-tuning, the evaluation methods for model validation, and our study results as well as the generalizibility of our results.

Retraining dataset: Our retraining dataset contains 7 million words, which is comparable with the retraining dataset used for

SCIBERT (1.14 million words) [15]. However, other domain-specific LMs are retrained using a substantially larger dataset than ours, with BiOBERT [14] training on 13.5 billion words, BERTweet [22] on 850 million words and LEGAL-BERT [16] on 6 billion words. Nonetheless, we believe that our retraining dataset does not pose a serious threat to our research results.

Fine-tuning dataset: Due to the unavailability of the labelled datasets suitable for our concept extraction task, we developed a fine-tuning dataset ourselves. To minimize the bias, we composed the dataset using the requirements documents from different domains (Section 3.21). To ensure the concepts are clearly identifiable, we selected only functional requirements for labelling. To ensure the labelling reliability, we recruited two requirements engineers with extensive domain knowledge to independently label the dataset and resolve any disagreements on the labelling results through discussion. To ensure the annotators have a common understanding of the concepts to be labelled, we provided a clear annotation scheme by defining the different classes of semantic roles and relation tags used for labelling. In spite of our efforts, the labelled dataset is limited in its size, as it only contains 1055 labelled sentences with a maximum sentence length of 90. Producing more labelled datasets for RE tasks remains to be an open challenge in using supervised learning methods in RE [31].

Experimental design: One potential internal validity threat concerns our experimental design, to mitigate this threat, we followed the experimental design guidelines by Kitchenham et al. [38]. For both the retraining and fine-tuning experiments, we ensured that our experiment procedures were clearly explained and design choices justified. We therefore believe there is no serious threat to our experimental design.

Evaluation method: As our dataset is small, we did not validate our BERT4RE and $\text{BERT}_{\text{base}}$ models using standard **k-fold** cross validation [39]; instead, we adopted a common hold-out method, by splitting the dataset into 70-30, using 70% of the data for training and the remaining 30% of the data for testing. We measured the performance of our models using the standard precision, recall and F1-score metrics. We noticed that other RE researchers also applied the project-level specific cross validation methods, such as **p-fold** [30] and leave-one-project-out [12], to their machine learning model. These project specific validation methods can ensure the model to be thoroughly validated, by exposing it to different aspects of the data in different combinations. However, we found that these methods would only be beneficial if the dataset is large; for a small dataset like ours, these methods (including k-fold) would only make the dataset too fragmented and reduce the performance of the model.

Results limitation and generalizibility: The results obtained in our research are based on one single dataset with a limited amount of labelled requirements text. In addition, the dataset was collected from the publicly available requirements documents crawled online [31]. Consequently, we cannot be sure if such results are repeatable on different datasets, particularly the datasets from the industry standard projects. To address this threat, we are making our BERT4RE model available [18] for RE researchers and practitioners to try and we welcome feedback from you.

7. Conclusion and Future Plan

In this paper, we present BERT4RE, a retrained BERT model for RE. BERT4RE is retrained on a RE related dataset, intended to be transferable to performing different RE specific tasks through model fine-tuning. As the initial step, this paper demonstrates the transferability of BERT4RE on the task of identifying 9 requirements concepts. To assess its effectiveness, we compare its performance with that of $\text{BERT}_{\text{base}}$ by fine-tuning the both models using a labelled dataset with nine classes. Our study shows that BERT4RE outperforms $\text{BERT}_{\text{base}}$ on identifying 7 out of 9 concepts and thus demonstrates the benefit from retraining the LM.

To take this research forward, we intend to conduct the following four projects:

The first project involves demonstrating the effectiveness of using domain-specific LM with transfer learning ability on a wide range of RE tasks. We intend to fine-tune BERT4RE for other RE tasks such as classification using the PROMISE dataset, traceability and detection of language issues among others.

The second project is the exploration of how effective our proposed technique is for the development of automatic tools that support the identification of key concepts in requirements text. By defining a modelling framework, the BERT4RE model will be extended to identify key concepts and create models that represents requirements text as a first class artefacts. The framework will be conceptualized as a semantic approach that identifies *semantic roles* of syntactic features and transform these roles (such as agents, actions, goals etc.) and their relationships into a conceptual model.

The third project will be building on the process and approach used in developing the BERT4RE model. We want to make our model robust by upgrading its architecture to train other advanced LM model variants such as ALBERT [40] where a two-parameter reduction approach is used to lower memory consumption and increase training speed of BERT. This approach we believe will enable the training and scaling of domain-specific models such as our BERT4RE model. Also, another LM we are adopting its architecture is the ELECTRA LM [41]. The architecture allow us to be able to train a discriminate model that in contrast to the BERT model [2] that only works with the masking of input token. We want to use a sample-efficient approach that seek to replace token detection rather than masking input tokens. This approach we envisage enables the definition of pre-training task on our input tokens instead of masking small input subset.

The fourth project of this research will seek to upscale and validate the performance of our BERT4RE model. This will be conducted through the fine-tuning of domain-specific data and specifically legal and social-media related data as case studies. The fine-tuning shall be purposed towards the extraction of semantic information through several downstream NLP tasks.

This will believe will help address the lack of training data in a domain-specific field like RE.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[4] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf, 2018.

[5] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[7] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings," *arXiv preprint arXiv:1909.00512*, 2019.

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[10] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[11] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–41, 2021.

[12] T. Hey, J. Keim, A. Koziolek, and W. Tichy, "Norbert: Transfer learning for requirements classification," *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 169–179, 2020.

[13] A. Sainani, P. R. Anish, V. N. Joshi, and S. Ghaisas, "Extracting and classifying requirements from software engineering contracts," *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 147–157, 2020.

[14] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[15] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *arXiv preprint arXiv:1903.10676*, 2019.

[16] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," *arXiv preprint arXiv:2010.02559*, 2020.

[17] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[18] M. Ajagbe and L. Zhao, "Supplementary material of "retraining a bert model for transfer learning in requirements engineering: A preliminary study"," March 2022. [Online]. Available: https://zenodo.org/record/6354280

[19] E. Alsentzer, J. Murphy, W. Boag, W. Weng, D. Jin, T. Naumann, and M. B. A. McDermott, "Publicly available clinical bert embeddings," *ArXiv*, vol. abs/1904.03323, 2019.

[20] K. Huang, J. Altosaar, and R. Ranganath, "Clinicalbert: Modeling clinical notes and predicting hospital readmission," *ArXiv*, vol. abs/1904.05342, 2019.

[21] A. E. W. Johnson, T. Pollard, L. Shen, L. wei H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Celi, and R. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, vol. 3, 2016.

[22] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "Bertweet: A pre-trained language model for english tweets," *arXiv preprint arXiv:2005.10200*, 2020.

[23] M. Li, L. Shi, Y. Yang, and Q. Wang, "A deep multitask learning approach for requirements discovery and annotation from open forum," *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 336–348, 2020.

[24] Y. Wang, L. Shi, M. Li, Q. Wang, and Y. Yang, "A deep context-wise method for coreference detection in natural language requirements," *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 180–191, 2020.

[25] A. F. Araujo and R. M. Marcacini, "Re-bert: automatic extraction of software requirements from app reviews using bert language model," *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021.

[26] M. Pota, M. Ventura, R. Catelli, and M. Esposito, "An effective bert-based pipeline for twitter sentiment analysis: A case study in italian," *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[27] J. Cleland-Huang, S. Mazrouee, H. Liguo, and D. Port, "NFR," Mar 2007. [Online]. Available: https://doi.org/10.5281/zenodo.268542

[28] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements engineering*, vol. 12, no. 2, pp. 103–120, 2007.

[29] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 490–495.

[30] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 142–152.

[31] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 502–505.

[32] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[35] C. J. Fillmore *et al.*, "Frame semantics and the nature of language," in *Annals of the New York Academy of Sciences: Conference on the origin and development of language and speech*, vol. 280, no. 1. New York, 1976, pp. 20–32.

[36] K. Letsholo, L. Zhao, and E.-V. Chioasca, "Tram: A tool for transforming textual requirements into analysis models," *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 738–741, 2013.

[37] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[38] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on software engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[39] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation." *Encyclopedia of database systems*, vol. 5, pp. 532–538, 2009.

[40] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[41] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.