NoRBERT: Transfer Learning for Requirements Classification

Tobias Hey, Jan Keim, Anne Koziolek, Walter F. Tichy
Karlsruhe Institute of Technology (KIT)
Institute for Program Structures and Data Organization
Karlsruhe, Germany
hey@kit.edu, jan.keim@kit.edu, koziolek@kit.edu, tichy@kit.edu

***Abstract*—Classifying requirements is crucial for automatically handling natural language requirements. The performance of existing automatic classification approaches diminishes when applied to unseen projects because requirements usually vary in wording and style. The main problem is poor generalization. We propose NoRBERT that fine-tunes BERT, a language model that has proven useful for transfer learning. We apply our approach to different tasks in the domain of requirements classification. We achieve similar or better results (F_1-scores of up to 94%) on both seen and unseen projects for classifying functional and non-functional requirements on the PROMISE NFR dataset. NoRBERT outperforms recent approaches at classifying non-functional requirements subclasses. The most frequent classes are classified with an average F_1-score of 87%. In an unseen project setup on a relabeled PROMISE NFR dataset, our approach achieves an improvement of 15 percentage points in average F_1-score compared to recent approaches. Additionally, we propose to classify functional requirements according to the included concerns, i.e., function, data, and behavior. We labeled the functional requirements in the PROMISE NFR dataset and applied our approach. NoRBERT achieves an F_1-score of up to 92%. Overall, NoRBERT improves requirements classification and can be applied to unseen projects with convincing results.**

***Index Terms*—Requirements Classification, Requirements Engineering, Machine Learning, Transfer Learning, Language Model, BERT**

I. Introduction

The main sources of requirements are still natural language documents. Classifying requirements is important to identify specific requirements like security-related requirements early in a project [1]. Furthermore, the automatic processing of natural language requirements also requires the identification of requirements in these documents. Automatic requirements processing approaches may benefit from a categorization of requirements into functional requirements (F), or quality requirements and constraints (together referred to as non-functional requirements (NFR)). Even though the distinction between functional and non-functional requirements is controversial [2]–[4], the need for categorization remains.

Automated classification approaches have been used for several years [5]–[8]. While they achieve reasonable performance on heterogeneous datasets, their performance diminishes when being applied to unseen projects [9]. State-of-the-art approaches use lexical and syntactical features, but still seem to lack the ability to generalize. Wording, sentence structure, and granularity of natural language requirements highly depend on the project and authors. Without transferability to unseen projects, current approaches are not applicable in practice. One would need a suitable training set for each project, which is usually infeasible. To overcome this challenge, we investigate how transfer learning approaches perform on the task of requirements classification. Transfer learning approaches are heavily used in natural language processing (NLP). They are trained on huge datasets to capture underlying concepts and meanings of natural language texts. Afterwards they can be fine-tuned to a specific task. These approaches promise both better performance and generalizability with less training data. Recent transfer-learning approaches were able to match performance with other (deep learning) approaches that were trained on 100x the data [10]. In cases such as in requirements engineering, where only limited amount of (labeled) data exists, this might prove advantageous.

We propose to fine-tune Bidirectional Encoder Representations from Transformers (BERT) [11], a language model based on deep learning. BERT, pre-trained on a large text corpus, can be fine-tuned on specific tasks by providing only a small amount of data. We present our approach NoRBERT (Non-functional and functional Requirements classification using BERT) that takes advantage of BERT's fine-tuning mechanism to classify requirements. We use NoRBERT to classify requirements in the PROMISE NFR dataset [12] widely used in literature and part of the RE'17 Data Challenge. Further, we investigate the generalizability of our approach on a relabeled version of the NFR dataset provided by Dalpiaz et al. [9]. We also use NoRBERT to classify functional requirements based on the implied concerns; a concern-based classification based on Glinz [2]. Our contribution is three-fold:

1) We investigate if and to what extent classifying requirements on known and unknown projects is improved by transfer learning.
2) We provide a new dataset that classifies functional requirements further according to the concerns function, data, and behavior.
3) We evaluate how well a transfer learning-based approach performs on the new dataset and task.

We provide our source code and the labeled functional requirements dataset on Zenodo [13].

II. Related Work

The distinction between functional and non-functional requirements and the definition of non-functional requirements itself is a well studied topic in requirements engineering. Requirements have been modeled as goals [14], [15] and non-functional requirements as softgoals, without a criterion for satisfaction. Li et al. [3] use a quality-oriented approach to model non-functional requirements. They regard non-functional requirements as either quality goals or quality constraints; the latter are perceivable or measurable. Glinz [2] classifies requirements as either functional requirements, system attributes, or constraints. System attributes or constraints are non-functional. Others argue that the distinction between functional and non-functional requirements is artificial [16] and many non-functional requirements also include functional aspects [4].

The automated extraction and classification of requirements from textual natural language documents has been in the focus of researchers for more than a decade resulting in many approaches. Approaches by Cleland-Huang et al. [5], [17] use information retrieval to classify non-functional requirements in structured and unstructured documents. They identify so-called "indicator terms" in the documents and use them to classify requirements. The approach achieves high recall values but is imprecise (precision below 27%) when classifying non-functional requirements. They also published their dataset NFR [12], that we use in our evaluation, as part of the PROMISE repository [18].

Hussain et al. [6] use special classes of words, such as cardinals, adverbs, and modal verbs with a decision tree classifier (C4.5) to improve requirements classification on a different version of the PROMISE NFR dataset, achieving promising results (up to 99% F_1-score).

Kurtanović and Maleej [8] use automated feature selection on lexical, syntactical, and meta-data features to predict certain classes of requirements in the PROMISE NFR dataset. Their support vector machine classifier achieves F_1-scores of up to 93% for functional vs. non-functional classification. On the four major non-functional requirement classes Usability, Security, Operational and Performance, their binary classifiers achieve results ranging between 72% and 90%. We also measure NoRBERT's performance on both of these tasks.

Abad et al. [7] show that preprocessing and unifying the PROMISE NFR dataset improves the result of a decision tree classifier, like the one proposed by Hussain et al. [6], on the functional vs. non-functional classification from F_1-score of 91% to 95%. Furthermore, they evaluate several classification algorithms on the task of classifying non-functional requirement subclasses that we also perform. The binarized naïve Bayes classifier performs best on both the unprocessed and preprocessed dataset achieving F_1-score of 45% and 90%, respectively. One drawback of this approach is the partially manual preprocessing that might be dataset specific.

Dalpiaz et al. [9] relabel the PROMISE NFR dataset to fix labeling issues and take requirements that comprise functional and non-functional aspects into account. They reimplement the approach of Kurtanović and Maleej [8] and evaluate it on the relabeled dataset and further projects. Additionally, they propose a more interpretable feature set achieving lower but comparable results on the task. In our evaluation, we use their dataset for the sake of comparability.

Other approaches use deep learning to classify requirements. For example, Winkler and Vogelsang [19] use a convolutional neural network (CNN) to classify the content elements of natural language requirement specifications as either "requirement" or "information" achieving F_1-scores of 82%. Navarro-Almanza et al. [20] employ a CNN to classify the requirements in the PROMISE NFR dataset according to all twelve requirement categories with an average F_1-score of 77%. Dekhtyar and Fong [21] apply word embeddings and a CNN to the task of identifying non-functional requirements in the PROMISE NFR dataset. They achieve an F_1-score of up to 92%. Amasaki and Leelaprute [22] compare different word vector models in terms of their effectiveness on the four major non-functional requirement classes in the PROMISE NFR dataset. The results show that Doc2Vec [23] and sparse composite document vector [24] based models can improve the classification of non-functional requirement categories depending on the classifier.

The above approaches are promising and show the capabilities of different techniques for the problem of classifying requirements. However, many of these approaches are rather impractical in use, as they are either overfitted to the dataset, heavily depending on wording and sentence structure, or need manual preprocessing. Moreover, the approaches either do not state their abilities to generalize from project specifics or do not generalize well enough to be applicable to previously unseen projects. One factor might be the lack of available training data in the requirements engineering community. This is the reason why we apply transfer learning approaches to requirements classification that promise better performance and generalizability with less training data.

III. Introduction to BERT

BERT [11] is a language model (LM). LMs aim to estimate the probabilities of sequences of words, thus are able to predict the probability that a word follows a given sequence. One important aspect of LMs like BERT are their transfer learning capabilities, i.e., using them for tasks other than the task they were trained on with little fine-tuning effort.

The origin of all modern LMs are word embeddings such as *word2vec* [25]. Mikolov et al. introduce a technique to calculate lower-dimensional vectors that represent words as numerical vectors based on their contexts. However, word2vec averages all contexts a word can appear in and thus disregards ambiguities. This problem is tackled by *Embeddings from Language Models (ELMo)* [26] that contextualizes word embeddings. ELMo is trained using a bidirectional long short-term memory (LSTM) neural network with two layers in order to let the LM get a sense of the word in the context of its previous and following words. Based on ELMo, *Universal Language Model Fine-tuning for Text Classification (ULMFiT)* improves LMs

further [10] and introduces pre-training and fine-tuning for transfer learning with LMs, that BERT uses as well.

An integral part of BERT is the so-called *Transformer* architecture by Vaswani et al. [27] that introduces a stacked self-attention encoder-decoder structure as a replacement for the LSTM architectures [28]. Besides the high-level encoder-decoder structure, the so-called *(self-) attention layers* allow to weight relevant words higher. For example, coreferences are highly relevant to each other and therefore are weighted higher than mostly unrelated words. This way, irrelevant words gain less attention and influence the outcome less than relevant words. Vaswani et al. [27] also introduce *multi-headed attention* as an improvement to previous attention models and mechanisms. The multi-headed attention helps the overall model to focus on different positions and solves the problem that the current word itself can dominate other words.

Radford et al. [29] introduce the concepts of the Transformer architecture to LMs that can be fine-tuned. They use the decoder structure of the Transformer, including the attention layers, for language modeling. After pre-training, the model can be used for downstream NLP tasks such as sequence classification.

The BERT model by Devlin et al. [11] combines various of the previously described concepts to train LMs. On release BERT outperformed state-of-the-art results on eleven NLP tasks. Benchmarks include sequence classification, named entity recognition, and question answering [11]. In contrast to former Transformer-based approaches that only use unidirectional, i.e, left-to-right architectures, BERT introduces bidirectional pre-training similar to the concepts of ELMo to incorporate context in both directions. Instead of using the decoder-structure of Radford et al. [29], Devlin et al. adapt the encoder-structure of the Transformer architecture by Vaswani et al [27]. They propose two models for BERT. The base model has twelve encoder-layers, uses 768 hidden units, and twelve attention heads [11] instead of the original Transformer's six encoder-layers, 512 hidden units, and eight attention heads [27]. BERT's base model has a total of 110M parameters that need to be trained. The large model of BERT uses 24 encoder-layers with encoders having 1024 hidden units, 16 attention heads and 340M parameters in total.

The combination of encoders and self-attention layers with bidirectionality causes a problem. Each word indirectly sees itself and the target word, during training, and could be trivially predicted. Therefore, the authors introduce *masked* language modeling. This means that about 15% of the input is masked and part of the pre-training task is the prediction of the masked words. For example, instead of the sentence "my dog is hairy" the input is "my dog is [MASK]." However, BERT was not only trained to predict missing words but simultaneously predicts whether one sentence is likely to succeed another sentence. For example, the sentence "The man went to the store" is likely to be followed by "He bought a gallon of milk" but unlikely to be followed by "Penguins are flightless birds." For training, 50% of the sentences were paired with actual subsequent sentences and 50% of the time with a random sentence from the corpus. BERT was originally trained on the English Wikipedia and the BooksCorpus [30]. Pre-trained models of BERT have been released that can be used for fine-tuning.

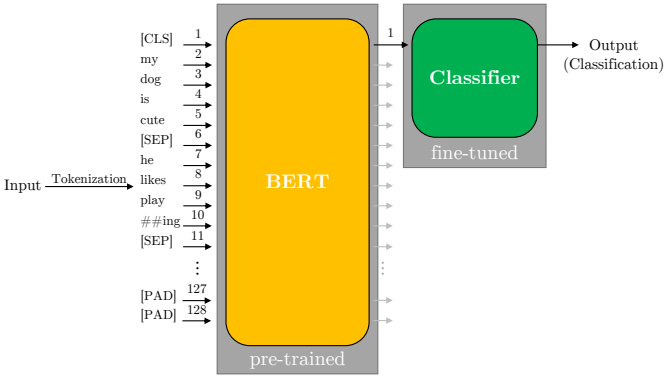

Fig. 1: Architecture used to fine-tune BERT for classification.

BERT and similar approaches are currently replacing traditional pipelines as natural language processing systems [31]. However, an analysis of the different layers and the underlying learned structures of BERT by Tenney et al. [31] shows that BERT remodels similar structures as the NLP pipeline.

Figure 1 shows how BERT can be used for classification. The input is tokenized. The first input token of BERT is always the special token [CLS]. Similarly, the token [SEP] is a special separator token, e.g., to separate sentences, and the token [PAD] is used for padding. For classification and similar downstream tasks the only output of BERT used is the output BERT produces for the first token ([CLS]), which is the pooled output of all tokens. This pooled output can be fed into a single-layer feedforward neural network that uses softmax to assign probabilities to different classes.

IV. Research Design

First, we formulate the questions our research is going to answer and then describe the datasets and methodologies used.

A. Research Questions

RQ1: *How does transfer learning perform in classifying requirements?*

We want to investigate how an approach that uses transfer learning performs on the classical tasks of requirements classification.

RQ2: *Does transfer learning improve the performance of classifying requirements on unseen projects?*

As the performance of current approaches diminishes when applied to unseen projects, we want to study how a transfer learning-based approach performs at requirements classification on unseen projects.

RQ3: *To what extent do transfer learning approaches detect subclasses of functional requirements?*

We investigate if and to what extent classification models using transfer learning are able to identify the concerns described in functional requirements.

TABLE I: Class distribution of the original NFR dataset [12].

Class	Quantity	∅ Words
Functional (F)	255	20
Availability (A)	21	19
Fault Tolerance (FT)	10	19
Legal (L)	13	18
Look & Feel (LF)	38	20
Maintainability (MN)	17	28
Operational (O)	62	20
Performance (PE)	54	22
Portability (PO)	1	14
Scalability (SC)	21	18
Security (SE)	66	20
Usability (US)	67	22
Total	625	20
Non-Functional (NFR)	370	20

B. Existing Datasets

To answer RQ1 and RQ2 we make use of two existing datasets. The PROMISE NFR dataset [12] commonly used in the community and addressed in the RE'17 Data Challenge and a relabeled version provided by Dalpiaz et al. [9]. The former consists of 625 requirements from 15 projects written by students. The 625 requirements include 255 functional and 370 non-functional requirements. Table I shows the distribution of classes in the dataset and the average length of the requirements per class. Each requirement is labeled with only one class (either F or one of the 11 NFR subclasses). The classes are distributed unevenly. Tha dataset contains 115 less F than NFR and the quantity of NFR subclasses differs widely, ranging from 67 for Usability to 1 for Portability. The classes Usability, Security, Operational, and Performance are the only ones exceeding 50 examples, whereas the classes Fault Tolerance, Legal, Maintainability and Portability are underrepresented with amounts below 20.

As the distinction between F and NFR in the dataset is debatable and the dataset includes duplicates and mislabeled requirements, Dalpiaz et al. [9] provided a relabeled version of the dataset. Table II presents an overview of the dataset. It consists of only 612 of the original 625 requirements and only uses two classes. They followed the quality-oriented approach by Li et al. [3] to model the classes. A requirement can have functional (F) or quality aspects (Q) or both. 80 requirements include both functional and quality aspects. 230 requirements solely have functional aspects (OnlyF) and 302 include only quality aspects (OnlyQ).

C. Research Methodology

Based on the datasets we apply our approach on the tasks:

Task1: Binary classification of F/NFR on the original NFR dataset. We collapsed all NFR subclasses to represent the NFR class.

Task2: Binary and multiclass classification of the four most frequent NFR subclasses (US, SE, O, PE) on all NFR in the original NFR dataset.

Task3: Multiclass classification of all NFR subclasses on NFR in the original NFR dataset.

TABLE II: Class distribution of the relabeled NFR dataset [9].

Class	Quantity	∅ Words
Functional aspect (F)	310	19
Quality aspect (Q)	382	20
Only Functional aspect (OnlyF)	230	19
Only Quality aspect (OnlyQ)	302	20
Both (F+Q)	80	21
Total	612	20

Task4: Binary classification of requirements based on functional and quality aspects using the relabeled NFR dataset by Dalpiaz et al.

For all tasks we measure precision (P), recall (R) and F_1-score (F_1). For the multiclass classifications we also report the weighted average F_1-score (A) over predicted classes.

The tasks are evaluated in various settings. With **.75-split** we describe a single stratified 75% train and 25% test split of the dataset. We also use a stratified 10-fold cross-validation referred to as **10-fold**, splitting the dataset 10 times in 90% train and 10% test set and averaging the results. Stratified splits ensure that the distribution of classes in the dataset is maintained in the train and test set. To further investigate the transferability of approaches we use two project-specific folding strategies. With **p-fold** we describe the project-level cross-validation used by Dalpiaz et al. [9], that splits the dataset 10 times in 3 projects as test and 12 projects as train set ensuring an even distribution of functional and quality aspects. Furthermore, we make use of a leave-one-project-out cross-validation (**loPo**) that n times trains on n-1 projects and tests on the project that was left out.

For highly imbalanced binary tasks, such as the NFR subclasses, we experimented with under- and oversampling strategies. We randomly sample a number of majority class representatives equal to the number of minority class examples in the training set for undersampling (US) the majority class. With oversampling (OS), we repeatedly add the whole training set minority class population to the training set until the number of minority class representatives would exceed the one of the majority class. Then we add random samples of the minority class until the classes are evenly distributed in the training set. Thus, US reduces the majority population and leaves the minority class untouched, whereas OS enlarges the minority population and keeps the original majority population. The test set remains untouched in both settings to maintain the distribution in the whole dataset.

We investigate the effect of early stopping (ES) and different epoch numbers. Early stopping is a regularization technique commonly used to avoid overfitting in iterative learners. We defined a threshold of 0.01 on the F_1-score of the class to predict (binary case) or accuracy (multiclass setting) and a patience value of three. The training is stopped if the improvement to the best iteration is three iterations (epochs) below the threshold.

V. NoRBERT: Non-functional and functional Requirements classification using BERT

With a fine-tuned version of BERT we investigate the impact of transfer learning on the task of requirements classification.

TABLE III: F/NFR classification on PROMISE NFR dataset. Bold values show the highest score for each metric per class.

Approach (Parameters)	F (255) P	F (255) R	F (255) F_1	NFR (370) P	NFR (370) R	NFR (370) F_1
10-fold						
K. & M. (word features w/o feat. sel.)	**.92**	.93	**.93**	.93	.92	.92
K. & M. (500 best word features)	**.92**	.79	.85	.82	.93	.87
K. & M. (500 best features)	.88	.87	.87	.87	.88	.87
A. et al. (unprocessed data)	.84	.93	.88	.95	.88	.91
A. et al. (processed data)	.90	**.97**	**.93**	**.98**	.93	**.95**
D. & F. (word2vec, ep.=100, f.=50)	—	—	—	.93	.92	.92
NoRBERT (base, ep.=10)	.91	.90	.90	.93	.94	.93
NoRBERT (base, ep.=10, ES, US)	.88	.88	.88	.92	.92	.92
NoRBERT (base, ep.=10, ES, OS)	.91	.86	.88	.91	.94	.92
NoRBERT (base, ep.=16)	.89	.88	.89	.92	.93	.92
NoRBERT (large, ep.=10, OS)	**.92**	.88	.90	.92	**.95**	.93
p-fold						
NoRBERT (base, ep.=10)	.91	.86	.88	.91	.94	.92
NoRBERT (large, ep.=10)	**.93**	**.88**	**.91**	**.92**	**.95**	**.94**
loPo						
NoRBERT (base, ep.=10, ES)	.91	.88	**.90**	**.92**	.94	.93
NoRBERT (large, ep.=16)	**.92**	**.89**	**.90**	**.92**	**.95**	**.94**

The rationale behind this decision is our expectation that models based on BERT generalize better on less training data and thus increase classification performance on unseen projects. For fine-tuning, we use two different pre-trained BERT models, the base and large model, both in the cased version. We also experimented with the uncased models but the cased models outperformed them. This might be due to named entities used in requirements that are mistaken for normal nouns otherwise. We use the BERT-tokenizer and do not preprocess the requirements.

On top of the pre-trained models we define our output layer, the classification head. We use the pooled output of BERT, i.e., the output of the [CLS] token, the first token in the sequence. This output is fed into the classification head that consists of a single layer of linear neurons in a feedforward neural network. The outputs are directly computed from the sum of the weighted inputs (plus some bias). We use the softmax function to get a probability distribution for the different labels.

During training, we use the cross-entropy loss function. This means we quantify how close the predicted distribution is to the true distribution using the formula:

$$H(p,q) = -\sum_{x} p(x) \log q(x) \tag{1}$$

$p(x)$ represents the target probability and $q(x)$ is the actual probability; x represents the different labels. In our case, for a correct label x, $p(x)$ equals 1 and $p(x)$ equals 0 for all other labels x. Thus, the loss function punishes wrong or uncertain predictions and rewards confident predictions that are correct.

An important component of (deep) neural networks is the optimizer that updates the various weights within a network. Instead of using a classical stochastic gradient descent to update the weights in our network, we use the so-called *AdamW*-optimizer that is an adaptation of the popular *Adam*-optimizer [32]. AdamW [33] implements a weight decay correction and does not compensate for bias as in the regular Adam-optimizer. As optimizer setting, we do not use a warm-up phase to increase the performance on sparse datasets. We use a weight decay of 0.01 and a maximal learning rate of 2e-05 also used in the original BERT publication [11]. Experiments showed that the standard batch size of 16 performed best throughout all runs (we also tried 14, 20, 32, 64). We use a maximal sequence length of 128 for the base models and 50 for the large models. Lowering the sequence length allows us to optimize performance and avoid memory issues. Overall, less than 10 requirements exceed a sequence length of 50. Therefore, we assume our choice to be a reasonable trade-off.

While fine-tuning NoRBERT's hyperparameters, i.e., the epoch number, we follow the rationale that higher epoch numbers allow the classifier to fit more closely to the seen data but poses the risk of overfitting. The epoch number defines the number of times the entire training set is processed during training (iterations of the iterative learner). Systematically increasing epoch numbers in our experiments showed that 10 to 32 epochs for the binary settings and 10 to 64 for multiclass setting performed best on the task. We cannot report all available evaluation results for the sake of brevity. Further results and the source code can be found on Zenodo [13].

VI. Task1: Classifying Functional and Non-Functional Requirements

For the first task, we want to measure the performance of NoRBERT when classifying requirements as either functional (F) or non-functional (NFR) on the original PROMISE NFR dataset. We use the stratified 10-fold cross-validation setting to answer RQ1 regarding the performance of transfer learning for requirements classification. We trained the model for binary classification, i.e., predicting whether a requirement is F or NFR. We compare our results to the state-of-the-art approaches by Kurtanović and Maleej [8], Abad et al. [7] and Dekhtyar and Fong [21]. Table III shows our results in comparison to the reported results of the other approaches. NoRBERT achieves comparable results with an F_1-score of 90% for functional and 93% for non-functional requirements. On NFR, NoRBERT outperforms all but the highest scoring approach by Abad et al. [7], which depends on manually provided dictionaries and rules to preprocess the dataset. NoRBERT in comparison needs no manual preprocessing and thus can easily be transferred to any other dataset. On F, Kurtanović and Maleej report a higher F_1-score using a model without feature selection that only uses word features. They mention that this model overfits to the dataset and thus might not be applicable to unseen projects.

We report results on the 10-fold cross-validation using different parameters and settings. Undersampling (US) the majority class NFR as well as oversampling (OS) the minority class F does not improve the model although there are roughly 45% more NFR. The selection of the BERT model or the epoch number does not significantly impact the overall performance.

The results in Table III also contribute to RQ2 on the generalizability of our approach. 10-fold is common practice for classification tasks but does not take into account that the dataset consists of different projects with different domains and wording. To measure NoRBERT's performance on unseen

TABLE IV: Binary (bin) and multiclass classification (multi) of the most frequent NFR classes on NFR dataset. Bold values represent the highest score per metric per class. Asterisks mark reported F_1-scores that do not match precision and recall.

			Usability (US)			**Security (SE)**			**Operational (O)**			**Performance (PE)**			
	Approach	Parameters	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	A
10-fold	K. & M.$_{bin}$	(w/o feature selection)	.81	.85	.82	.91	.90	*.88	.72	.75	.73	.93	**.90**	***.90**	.83
	K. & M.$_{bin}$	(50 best features)	.70	.57	.61	.81	.77	*.74	.78	.50	*.57	.87	.57	.67	.65
	K. & M.$_{bin}$	(500 best features)	.80	.71	.74	.74	.81	*.74	.72	.73	*.71	.87	.81	*.82	.75
	NoRBERT$_{bin}$	(base, ep.=10)	.81	.69	.74	**.93**	.82	.87	.80	.53	.64	.88	.80	.83	.77
	NoRBERT$_{bin}$	(base, ep.=10, OS)	.78	.70	.74	.90	.86	.88	.88	.71	.79	.88	.80	.83	.81
	NoRBERT$_{bin}$	(base, ep.=16, OS, ES)	**.89**	.70	.78	.89	.89	.89	**.90**	.71	.79	.88	.81	.85	.83
	K. & M.$_{multi}$	(w/o feature selection)	.65	.82	*.70	.81	.77	*.75	.81	.86	.82	.86	.81	*.80	.76
	K. & M.$_{multi}$	(50 best features)	.49	.68	.55	.60	.50	*.39	.42	.47	*.33	.85	.53	*.63	.47
	K. & M.$_{multi}$	(500 best features)	.70	.66	*.64	.64	.53	.56	.47	.62	*.51	.81	.74	.76	.61
	NoRBERT$_{multi}$	(base, ep.=32)	.78	.84	.81	.89	.85	.87	.79	.73	.76	.88	.78	.82	.82
	NoRBERT$_{multi}$	(large, ep.=32)	.86	.82	.84	.91	.91	**.91**	.83	.71	.77	.90	.81	.85	.84
	NoRBERT$_{multi_all}$	(base, ep.=16, ES)	.86	**.91**	**.88**	.77	**.92**	.84	.77	.79	.78	.87	.83	.85	.84
	NoRBERT$_{multi_all}$	(base, ep.=50)	.78	.85	.81	.78	**.92**	.85	.83	**.84**	**.83**	**.94**	.87	**.90**	.85
	NoRBERT$_{multi_all}$	(large, ep.=50)	.83	.88	.86	.90	**.92**	**.91**	.78	**.84**	.81	.92	.87	**.90**	**.87**
p-fold	NoRBERT$_{bin}$	(base, ep.=16, OS)	.75	.70	.73	.77	.83	.80	.72	.61	.66	.84	.50	.63	.71
	NoRBERT$_{bin}$	(large, ep.=16, OS)	.83	.75	.78	**.89**	.82	.85	**.85**	.71	**.77**	.88	.74	.80	.80
	NoRBERT$_{multi}$	(base, ep.=16)	.65	.72	.68	.77	.87	.82	.69	.63	.66	.81	.58	.68	.71
	NoRBERT$_{multi}$	(large, ep.=32)	**.84**	.80	**.82**	**.89**	.89	.89	.78	.70	.74	**.91**	**.80**	**.85**	**.83**
	NoRBERT$_{multi_all}$	(base, ep.=32)	.61	.77	.68	.81	.86	.83	.61	.84	.71	.80	.65	.72	.74
	NoRBERT$_{multi_all}$	(large, ep.=50)	.71	**.82**	.76	.88	**.92**	**.90**	.71	**.85**	**.77**	.85	.76	.80	.81
loPo	NoRBERT$_{bin}$	(base, ep.=16, OS)	.69	.72	.70	.80	.83	.81	.79	.73	.76	**.91**	.57	.70	.74
	NoRBERT$_{bin}$	(large, ep.=16, OS)	**.87**	.70	.78	.89	.82	.85	**.88**	.79	**.83**	.89	.78	.83	.82
	NoRBERT$_{multi}$	(base, ep.=16)	.65	.78	.71	.77	.89	.83	.67	.56	.61	.86	.57	.69	.71
	NoRBERT$_{multi}$	(large, ep.=32)	.85	.79	**.82**	**.92**	.91	**.91**	.80	.77	.79	.86	**.81**	**.84**	**.84**
	NoRBERT$_{multi_all}$	(base, ep.=32)	.61	**.81**	.70	.72	.88	.79	.68	.74	.71	.85	.72	.78	.74
	NoRBERT$_{multi_all}$	(large, ep.=50)	.65	**.81**	.72	.82	**.92**	.87	.66	**.84**	.74	.83	.63	.72	.76

projects, we use project-specific settings (loPo and p-fold). The results are similar or better than in the 10-fold evaluation. This indicates that NoRBERT is able to generalize from wordings seen during training. This is especially important as the best performing approaches in the 10-fold setting are either overfitted (K. & M.) or require manual dictionaries and rule definitions that have to be adapted to each project (Abad et al.). In contrast, NoRBERT requires no manual preprocessing.

The results of NoRBERT for binary classification of requirements into functional and non-functional are promising. The evaluation results are comparable to state-of-the-art approaches with the additional benefit of generalizability, as the approach can be applied to unseen projects with no decrease in performance.

VII. Classification of Non-Functional Requirements Subclasses

In this next evaluation, we want to tackle the task of classifying NFR subclasses defined in the PROMISE NFR dataset. We filter out all F from the dataset and perform multiple evaluations to compare our approach with the reported results of recent papers. Filtering out F is reasonable as Task1 shows that we are able to detect NFR with high precision. First, we look into classification of the four most frequent NFR in the dataset: Usability, Security, Operational, and Performance (Task2). We evaluate multiple binary classifiers, one for each class, as well as multiclass classification in Subsection VII-A. Additionally, we look into the classification of all NFR subclasses (Task3) in Subsection VII-B.

A. Task2: Classification of most frequent NFR subclasses

We evaluate the performance of NoRBERT using one binary classifier for each of the four most frequent NFR in the dataset, i.e., Usability, Security, Operational, and Performance. As neural network classifiers are known to struggle with highly imbalanced datasets, we also apply multiclass classification to this task. Therefore, we train multiclass classifier either on the four most frequent subclasses plus "Other", or on all NFR subclasses. Table IV reports the results and compares them to those of Kurtanović and Maleej [8]. Approaches with the suffix "all" denote classifiers trained on all NFR classes. The last column shows the weighted average F_1-score over all classes weighted by the frequency of appearance.

The results of NoRBERT are promising with a weighted average F_1-score of up to 83% for binary classification and 87% for multiclass classification in the 10-fold cross-validation. On average, NoRBERT outperforms the best results of Kurtanović and Maleej [8] by more than three percentage points.

The results of NoRBERT exceed the results of the feature selection approaches of [8] both in the binary and the multiclass versions. Compared to the models of Kurtanović and Maleej without feature selection, NoRBERT performs better on three of four classes US, SE and O (F_1-score of 88% for US, 91% for SE and 83% for O) and achieves similar results on the fourth (F_1-score of 88% for PE). However, [8] seems to have reported some faulty values as the reported values for precision and recall do not add up to the reported value for F_1-score. These cases are marked with asterisks. Therefore, we cannot draw a final conclusion for the comparison. Additionally, the

TABLE V: Multiclass classification of all NFR subclasses on NFR dataset. 16, 32 and 50 indicate the used epoch number, *bin* binary and *mult* multiclass classification and B and L the used BERT model (base/large). bin_{16} additionally uses OS and $multiL_{32}$ ES. LDA and NB (Naïve Bayes) refer to approaches by Abad et al. [7] with (P) or without (UP) preprocessed data.

		A (21)			**FT (10)**			**L (13)**			**LF (38)**			**MN (17)**			**O (62)**			**PE (54)**			**SC (21)**			**SE (66)**			**US (67)**			
	Model	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	A
10-fold	bin_{16}	**.93**	.62	.74	.50	.20	.29	.82	.69	.75	.79	.71	.75	.54	.41	.47	**.87**	.77	.82	.88	.78	.82	.70	.67	.68	.88	.88	.88	**.87**	.72	.79	.78
	$multiB_{16}$	.73	.76	.74	**1.0**	.20	.33	.91	.77	.83	.77	.79	.78	**.86**	.35	.50	.77	.79	.78	.87	.83	.85	.68	.71	.70	.77	**.92**	.84	.86	**.91**	**.88**	.79
	$multiB_{32}$	.75	.71	.73	.38	.30	.33	.91	.77	.83	.81	.79	.80	.60	.35	.44	.78	.82	.80	.90	.83	.87	**.76**	**.76**	**.76**	.84	**.92**	.88	.78	.87	.82	.79
	$multiB_{50}$	.77	**.81**	**.79**	.60	.30	.40	.91	.77	.83	.80	.74	.77	.70	.41	.52	.83	**.84**	**.83**	**.94**	.87	**.90**	.64	.67	.65	.78	**.92**	.85	.78	.85	.81	.80
	$multiL_{32}$	.70	.76	.73	.56	.50	.53	**.92**	**.85**	**.88**	**.82**	**.87**	**.85**	.58	.41	.48	.74	.77	.76	.91	**.89**	**.90**	.70	.67	.68	.86	.89	.87	.86	.85	.86	**.81**
	$multiL_{50}$	.80	.76	.78	.60	**.60**	**.60**	.91	.77	.83	.81	.79	.80	.62	**.47**	**.53**	.78	.84	.81	.92	.87	**.90**	**.76**	**.76**	**.76**	**.90**	**.92**	**.91**	.83	.88	.86	**.82**
5x5-fold	LDA(P)	.60	.95	.74	.02	.10	.03	.20	.47	.28	.85	.60	.70	.52	.70	.60	.70	.35	.47	.95	.70	.81	.57	.81	.70	.87	.87	.87	.76	.61	.68	.62
	NB(UP)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	.45
	NB(P)	**.90**	**.90**	**.90**	**.90**	**.97**	**.93**	**1.0**	.75	**.86**	**.94**	**.94**	**.94**	**.82**	**.90**	**.86**	**.91**	.78	**.84**	**1.0**	**.90**	**.95**	**.83**	**.83**	**.83**	**1.0**	**.97**	**.98**	.77	**.97**	**.86**	**.90**
	$multiB_{32}$	.73	.77	.75	.65	.26	.37	.80	**.82**	.81	.77	.75	.76	.54	.40	.46	.79	**.82**	.81	.86	.81	.83	.68	.73	.70	.83	.89	.86	**.79**	.83	.81	.78
p-fold	bin_{16}	**.85**	.79	**.81**	**.50**	.10	.17	.50	.27	.35	.74	.42	.54	.38	.18	.24	**.72**	.61	.66	.84	.50	.63	.66	.60	.63	.77	.83	.80	**.75**	.70	.73	.64
	$multiB_{32}$	.64	.69	.67	.00	.00	.00	.67	.31	.42	.64	.49	.55	**.55**	.32	.41	.61	.84	.71	.80	.65	.72	.68	**.62**	.65	.81	.86	.83	.61	.77	.68	.66
	$multiL_{50}$	.73	**.83**	.78	**.50**	**.25**	**.33**	**.79**	**.58**	**.67**	**.82**	**.67**	**.74**	.52	**.35**	**.42**	.71	**.85**	**.77**	**.85**	**.76**	**.80**	**.72**	**.62**	**.67**	**.88**	**.92**	**.90**	.71	**.82**	**.76**	**.76**
loPo	bin_{16}	.71	.57	.63	**1.0**	**.20**	**.33**	.83	.38	.53	.81	.45	.58	.44	.24	.31	**.79**	.73	**.76**	**.91**	.57	.70	.65	.52	.58	.80	.80	.80	**.69**	.72	.70	.67
	$multiB_{32}$	**.79**	.71	.75	.00	.00	.00	.40	.15	.22	.64	.47	.55	.36	.29	.32	.68	.74	.71	.85	**.72**	**.78**	.68	**.62**	.65	.72	.88	.79	.61	**.81**	.70	.66
	$multiL_{50}$	.78	**.86**	**.82**	.50	**.20**	.27	**.89**	**.62**	**.73**	**.85**	**.58**	**.69**	**.62**	**.47**	**.53**	.66	**.84**	.74	.83	.63	.72	**.76**	**.62**	**.68**	**.82**	**.92**	**.87**	.65	**.81**	**.72**	**.73**

model of Kurtanović and Maleej without feature selection might be overfitted as it is based on the same feature set as the one discussed in Section VI. For RQ1 we can say that transfer learning based approaches are able to outperform recent approaches in subclassing non-functional requirements.

Besides the comparison, Table IV shows the results for different settings of NoRBERT. The best binary classification results are optained with 16 epochs, oversampling (OS) and early stopping. Surprisingly, the multiclass approach performs better than the binary classifiers and achieves the best performance, both overall and for each individual class. In most scenarios, a greater number of output nodes (for multiclass classifiers) increases the complexity of the model and should result in poorer results [34]. In addition, NoRBERT performs better when trained on all 10 classes (multi_all). We attribute this effect to the possibility that differences are subtle and the difference between the four most frequent classes are similar to the differences to requirements that do not belong to one of the classes. More classes and examples of the classes help NoRBERT to learn subtle differences and characteristics of each individual class, thus, help to figure out which requirements do not belong to one of the four classes.

We use the p-fold and loPo setting again to answer RQ2, the question of NoRBERT's performance on unseen projects. For p-fold, the performance decreases slightly for all classes, with a drop in F_1-score ranging from one percentage point (SE) to six percentage points (US, O). The results of the loPo setting are similar to the p-fold for US and PE but increase on SE and O. The performance on SE and O even matches the best results in the 10-fold. One explanation might be the higher amount of training data per fold in the loPo setting. Overall, the multiclass classifiers outperform the binary classifiers on this task as well. Interestingly, the large model, trained on four classes only, performs better on these settings than the model trained on all classes. In these settings the best model achieves a weighted average F_1-score of 83% (p-fold) and 84% (loPo), respectively. This shows that NoRBERT generalizes well on this task. The results exceed expectations as NoRBERT performs similar or better than the models of Kurtanović and Maleej in a possibly harder setting (10-fold vs. project-specific fold).

NoRBERT's performance on this task illustrates that transfer learning might enable the requirements engineering community to build classifiers that perform comparably on unseen as well as known projects with only a small amount of training data. This might overcome one of the major drawbacks in requirements classification, the lack of training data.

B. Task3: Multiclass classification of all NFR subclasses

In this section, we investigate the performance of NoRBERT using multiclass classification on all NFR subclasses using the original NFR dataset. We filtered out Portability, as it had only one representative in the dataset. It cannot be in training and test set at the same time and is thus impossible to predict. Therefore, we evaluate on one requirement less than in the original dataset. As a baseline we use binary classifiers for each of the 10 classes. We again use different settings and compare our results to Abad et al. [7] in a five times 5-fold.

Table V shows the results. The multiclass classifier perform reasonably well. Outliers are foremost classes with a small number of representatives. This strengthens our hypothesis that training data evenly distributed over the classes is a major factor for the success of this approach. All multiclass models outperform the respective binary classifiers regarding average performance. The best multiclass model even outperforms the binary classifiers on all but one class. This underpins our assumption that the approach has its strength on evenly distributed data (for training) and difficulties on highly imbalanced datasets. The models based on BERT-large ($multiL_{32}$ and $multiL_{50}$) perform best on this task. We attribute this to the larger parameter space and its ability to better cope with linguistic subtleties needed to distinguish the classes. Interestingly,

TABLE VI: Binary classification of classes in relabeled PROMISE NFR dataset. Bold values represent the highest score for each metric per class. Asterisks mark F_1-scores not matching precision and recall reported by other publications.

			F (310)			**Q (382)**			**OnlyF (230)**			**OnlyQ (302)**		
	Approach	Parameters	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
.75-split	K. & M. reimpl.	(100 best features)	.80	.78	*.80	**.91**	.90	*.88	.86	.82	*.89	.86	.75	.81
	K. & M. reimpl.	(500 best features)	.82	.80	*.82	**.91**	.89	*.87	.87	.87	*.91	**.90**	.85	.87
	Dalpiaz et al.	(final 17 features)	.71	.76	.73	.77	.80	*.92	.77	.83	*.72	.71	.75	*.78
	NoRBERT	(base, ep.=10)	.86	**.88**	.87	.90	.96	.93	.89	**.98**	**.93**	.87	**.93**	**.90**
	NoRBERT	(large, ep.=10)	**.92**	**.88**	**.90**	**.91**	**.99**	**.95**	**.92**	.93	.92	.82	.87	.85
10-fold	K. & M. reimpl.	(100 best features)	.82	.74	.77	.82	.91	.86	.82	.67	.73	.79	.81	.80
	K. & M. reimpl.	(500 best features)	.76	.68	.71	.79	.87	.82	.77	.63	.68	.74	.80	.77
	NoRBERT	(base, ep.=10)	.87	.84	.86	.92	**.97**	**.94**	**.92**	.89	**.91**	.85	**.85**	.85
	NoRBERT	(base, ep.=10, ES)	**.89**	**.86**	**.88**	.91	.96	**.94**	.91	.86	.88	**.87**	**.85**	**.86**
	NoRBERT	(large, ep.=10)	.86	**.86**	.86	**.93**	.95	**.94**	.90	**.90**	.90	.83	.75	.79
p-fold	K. & M. reimpl.	(100 best features)	.81	.70	.74	.75	.92	.82	.82	.52	.62	.75	.80	.77
	K. & M. reimpl.	(500 best features)	.75	.60	.66	.71	.88	.78	.75	.48	.57	.68	.79	.73
	NoRBERT	(base, ep.=10)	**.87**	.86	.86	.91	.95	.93	**.92**	.87	.89	.84	.86	.85
	NoRBERT	(large, ep.=10)	**.87**	**.87**	**.87**	.92	**.95**	**.94**	.89	**.90**	**.90**	.84	.74	.78
	NoRBERT	(large, ep.=10, ES)	**.87**	.85	.86	**.93**	**.95**	**.94**	.89	**.90**	.89	**.85**	**.89**	**.87**
loPo	NoRBERT	(base, ep.=10)	.86	**.85**	**.86**	.91	**.96**	.93	.89	.85	.87	.84	.88	.86
	NoRBERT	(base, ep.=10, ES)	**.87**	**.85**	**.86**	**.92**	**.96**	**.94**	**.90**	**.87**	**.89**	.85	.87	.86
	NoRBERT	(large, ep.=10)	.86	.84	.85	.90	**.96**	.93	.89	.82	.85	**.86**	**.89**	**.88**

multiL$_{50}$ is also the same model performing best on Task2. NoRBERT outperforms the results of the convolutional neural network-based approach by Navarro-Almanza et al. [20] by at least 5 percentage points (F_1-score of 82% vs. 77%) indicating that transfer learning outperforms word embedding-based deep learning on the task. To compare NoRBERT to Abad et al. [7], we perform a similar five times 5-fold cross validation. NoRBERT outperforms all approaches except the naïve Bayes classifier requiring (manually) preprocessed data. Our approach does not need any manual preprocessing and thus can be applied to new projects with ease.

The project-specific folding strategies reveal problems on unseen projects, as the results plummet for some classes, i.e., Fault Tolerance (FT) and Legal (L). This is due to the fact that the classes are not evenly distributed over the projects. The result is not surprising as FT and L are the classes with the least overall representation. Apart from these cases, the results for the p-fold (weighted average F_1-score of 76%) are comparable to the results achieved by the binary classifiers in the 10-fold setting and are only six percentage points lower than the best performing model. This is promising as it depicts that our approach generalizes well even in settings with only few examples per class in training. In the loPo setting NoRBERT performs slightly worse than in the p-fold. A possible explanation is that in p-fold some projects are tested more than once and these projects might be easier to predict. In all settings the multiclass classifier and the large models outperform the binary classifier.

The results on this task illustrate that NoRBERT is able to identify underrepresented NFR subclasses even in set-ups with only small amounts of training data and applied to unseen projects. It outperforms all approaches that do not preprocess the data. Thus, NoRBERT poses a practicable alternative to state-of-the-art approaches that require manual data preprocessing.

VIII. Task4: Functional and Quality Aspects

The distinction between F and NFR is not always clear and some requirements include aspects of both. Therefore, we measure NoRBERT on the relabeled PROMISE NFR dataset provided by Dalpiaz et al. [9]. They also reimplemented the approach of Kurtanović and Maleej [8] and measured it on the relabeled corpus. This allows us to compare NoRBERT to both approaches. Additionally, Dalpiaz et al. provide results on a project-specific fold that enables comparability for RQ2. Table VI shows the results of binary classifiers trained on the relabeled set. We use the same set-ups as Dalpiaz et al., i.e., .75-split, 10-fold, p-fold, and random seed (42) to compare the approaches. Additionally, we evaluate the loPo setting. NoRBERT outperforms the other approaches in all these settings. Thus, the transfer learning approach clearly increases the performance for classifying requirements (RQ1).

On the .75-split, base and large model versions of NoRBERT perform better on different classes. For classes containing only functional or only quality aspects, NoRBERT performs better with the base model. For the other two classes, the large model shows better results. On 10-fold cross-validation our best model outperforms the best model of Dalpiaz et al. by ten percentage points on average. Especially on requirements that only include functional aspects NoRBERT excels (F_1-score of 91% in contrast to 73%).

The results on p-fold and loPo provide insights on NoRBERT's ability to be transferred to unseen projects. The results are similar or slightly better than the results on the 10-fold setting. In the p-fold setting, in which we can compare our approach to the (reimplemented) approach of Kurtanović and Maleej, NoRBERT handles unseen projects better than the competitor. On average NoRBERT exceeds the reported F_1-scores by 15 percentage points. At a paired t-test on the F_1-scores of the four classes the improvement is statistically significant with a p-value of 0.016. This outcome is interesting, as this is

TABLE VII: Overview of the functional requirements dataset.

	Project														
Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
Function	14	13	33	19	25	29	10	31	11	9	3	7	2	1	207
Data	6	3	13	6	7	2	5	3	2	5	0	1	2	2	57
Behavior	2	9	9	13	6	6	5	14	10	33	3	1	1	1	113
Requirements	19	22	44	26	37	31	15	38	17	43	3	8	3	3	309

the only task in which we are able to compare NoRBERT's performance on unseen projects directly to a competitor. The performance of the reimplemented approach of Kurtanović and Maleej decreases by five percentage points on average, when applied to unseen projects (p-fold). In contrast, NoRBERT's performance on p-fold is similar to the performance on 10-fold. This indicates that NoRBERT generalizes well and can be used in practice without retraining, which was our hypothesis in the first place. In our opinion the effect of applying classifiers on unseen projects should be measured more often when evaluating requirements classification approaches. In practice, one seldomly has sufficient labeled data from the same project and wording as well as sentence structure highly depend on the project. Furthermore, approaches that generalize better can already be applied to new projects at early stages.

IX. Classifying Functional Requirements

As NoRBERT has proven useful in classifying requirements in general and non-functional requirements in particular, we want to investigate its performance on classifying functional requirements as well. Previous models categorize functional requirements according to the part of the product they belong to [35], [36], such as user interface or business logic. Other models take a concern-based approach [2], including functional and behavioral concerns as well as data. If we want to develop systems that interpret functional requirements automatically, e.g., automated traceability or modeling systems, the subclasses of functional requirements are relevant for further processing. They define how the functional requirements might be implemented. Therefore, we adapt the concern-based model by Glinz [2] that enables us to interpret whether a functional requirement describes functions of the system, behavior the system displays or mere data and the data structures the system contains. More precisely, we use the following subclasses:

Function: *A function that a system shall perform.*

Example: The system shall allow a real estate agent to query MLS information.

Data: *A data item or data structure that shall be part of a system's state.*

Example: The audit report shall include the total number of recycled parts used in the estimate.

Behavior: *Behavior the system displays or reactions that are triggered by one or more stimuli.*

Example: If the shot was marked as a hit the product shall allow the offensive player to define a shot.

As requirements may include multiple concerns, the classes might be overlapping, as it is the case in "Only registered customers can purchase streaming movies". It contains both, function and behavior.

TABLE VIII: Binary classification of functional requirement subclasses with NoRBERT on new dataset, with 10-fold and loPo. Bold values represent the highest score for each metric per class. *b* and *l* stand for the base and large model, respectively.

	Function (207)			**Data (57)**			**Behavior (113)**		
Parameter	P	R	F_1	P	R	F_1	P	R	F_1
10-fold									
b, ep.=16	.86	.95	.90	**.81**	.51	**.62**	.83	.62	.71
b, ep.=16, OS	**.87**	**.96**	**.91**	.63	.51	.56	.88	.70	.78
b, ep.=16, US	**.87**	.82	.84	.28	**.68**	.40	.82	**.78**	**.80**
l, ep.=16	.85	.94	.89	.66	.44	.53	**.91**	.68	.78
l, ep.=16, OS	.86	.95	.90	.74	.51	.60	.86	.73	.79
l, ep.=32, OS	.86	.86	.86	.76	.46	.57	.82	.72	.76
loPo									
b, ep.=16	.83	.83	.83	.56	.26	.36	.65	.55	.60
b, ep.=16, OS	.83	.89	.86	**.69**	.35	.47	.67	.64	.65
b, ep.=32, OS	.84	.85	.84	.67	.46	.54	.68	.56	.61
l, ep.=16	.77	.86	.81	.53	.16	.24	**.88**	.73	**.80**
l, ep.=16, OS	.82	.86	.84	.75	.16	.26	.76	.57	.65
l, ep.=32, OS	**.88**	**.95**	**.92**	**.69**	**.47**	**.56**	.86	**.74**	**.80**

To answer RQ3, whether transfer learning approaches are able to identify the concerns in functional requirements, we manually labeled all requirements tagged as having functional aspects (F) in the relabeled version of the PROMISE NFR dataset provided by Dalpiaz et al. [9]. The 310 requirements were tagged by two of the authors independently. We calculate Krippendorff's α (Kα) [37] to measure the inter-annotator agreement. For the classes *Function* (Kα 0.803) and *Data* (Kα 0.814) the Kα values exceed the commonly accepted threshold of 0.8 [38]. On *Behavior* the Kα is 0.752 which still is a reasonable agreement that exceeds the lower bound of 0.66, as reported by Krippendorff [38]. After the annotators finished labeling, they solved their disagreements in discussions to provide a uniform gold standard. Table VII shows the distribution of the classes over the projects. Note that the number of requirements is lower than the number of class representatives per project as each requirement can contain multiple concerns. The total number of requirements is 309 instead of 310 because we removed one duplicate requirement from Project 3. Projects 11 to 14 contain a small number of requirements as they mainly consist of non-functional requirements. We provide the dataset on Zenodo [13].

A. Evaluating NoRBERT on the functional requirements dataset

With the labeled dataset, we can evaluate the performance of NoRBERT on classifying functional requirements. We train binary classification models with varying parameters. We do not train multiclass variants, because the requirements in the dataset might belong to more than one class. We would need a proper multiclass multilabel approach that is out of scope for this paper. Once again, we evaluated the binary classifier in different settings, i.e., 10-fold cross-validation and loPo. However, we do not evaluate in a p-fold fashion as it is not reasonably applicable. Projects 11 to 15 are underrepresented in the dataset, which would skew the results on p-fold.

The results shown in Table VIII are promising for the classes Function and Behavior and reasonable on the Data class. NoRBERT achieves an F_1-score of up to 91% for Function and 80% for Behavior in the 10-fold cross-validation. However, NoRBERT achieves only low recall for the Data class. This can be attributed to the lack of training data for this class and the imbalanced dataset. Interestingly, undersampling improves the result on Behavior but decreases the results on the other two classes. Oversampling improves Behavior and Data, but the best performance on Data (F_1-score of 62%) was achieved with no sampling at all.

On unseen projects in the loPo setting, NoRBERT performs similar to the 10-fold setting. The result on Behavior is similar and for Function it even increases slightly to an F_1-score of 92%. Only on the Data class the performance decreases to an F_1-score of 56%. Nevertheless, compared to 62% in 10-fold cross-validation, the result is still promising as it is the least represented class. Interestingly, on loPo the best large model outperforms the base models on all classes, whereas on 10-fold the base models perform best. We attribute this to the effect that the loPo setting requires better generalizability.

To answer RQ3, we can conclude that for this amount of data NoRBERT does perform reasonably well but could be improved with more training data. Nevertheless, the performance on Function and Behavior might already be able to improve approaches such as trace link recovery or automated modelling.

X. Threats to Validity

In this section, we discuss potential threats to validity of our research and experimental design.

Construct Validity: To mitigate potential risks to construct validity, we applied widely used experimental designs and metrics. We use common practice in ML to fine-tune hyperparameters by experimenting with different parameter configurations. To mitigate the risk of unsystematic trial and error, we systematically increased the epoch number based on the assumption that an optimum between training performance and overfitting exists. For reproducibility, we used a fixed seed for the random number generators. We used the same seed (42) as Dalpiaz et al. for the experiment with the relabeled PROMISE NFR dataset [9]. For all other experiments, we used the randomly selected number 904727489 as seed.

Internal Validity: Considering the creation of the functional concerns dataset, a potential threat might be the fact that the dataset was created with the approach in mind. Therefore, there might be a risk of bias. We publish our data to mitigate this risk, so everyone can reproduce our classifications and findings.

External Validity: To show the generalizability of our approach, we apply different evaluation techniques. The techniques loPo and p-fold indicate for generalizability. However, the projects in the dataset might not be general representatives for all kind of projects. Moreover, the datasets that we used have some further issues in regard to external validity. Some requirements are incorrectly labeled and the selection of the requirements for the dataset is biased. This results in imbalanced datasets that miss some kinds of requirements, e.g., safety. Additionally, the requirements were written by students and thus might not illustrate industry standards. Therefore, conclusions on the generalizability based on the dataset are not warranted. We used these datasets as they are well-known and accepted in the community and let us directly compare our results to other approaches. We provide our code and data for further replications to mitigate this threat.

Conclusion Validity: A threat to the statistical conclusion validity might arise from the use of imbalanced datasets. The used dataset varies widely in regard to the support for the different classes. This problem is generally known in the RE community. Unfortunately, there is not enough data to properly apply mitigation techniques such as undersampling.

XI. Conclusion

In this paper, we presented NoRBERT, an approach for requirements classification that uses the transfer learning capabilities of BERT. We particularly investigated its performance on unseen projects, as we believe such investigation is widely undervalued in research but most important for application in practice. Our evaluation included common requirements classification tasks such as (binary) classification of functional and non-functional requirements (Task1), binary and multiclass classification of the four most frequent NFR classes (Task2), multiclass classification of all NFR classes (Task3), and binary classification of functional and quality aspects (Task4). We compared our results to state-of-the-art approaches and can conclude that NoRBERT outperforms all approaches that do not require manual preprocessing on Task2, Task3 and Task4 and performs similarly on Task1. The ability of transfer learning approaches manifests especially in project-specific folding strategies and thus when applying NoRBERT to unseen projects. On Task4 NoRBERT exceeds state-of-the-art results by 15 percentage points on average. This gives us the confidence to state that transfer learning approaches such as NoRBERT pose new capabilities to apply RE research in practice.

Additionally, we presented a novel requirements classification task that investigates the concerns of functional requirements. We see this task as an important step towards automatic interpretation of the purposes of functional requirements. We labeled the functional part of the NFR dataset according to the classes Function, Data, and Behavior. NoRBERT achieves F_1-scores of up to 92% at classifying these classes.

Overall, NoRBERT is a novel approach to classify requirements that outperforms recent approaches on most tasks. Especially on more realistic set-ups (unseen projects) it performs better (statistically significant at the 0.05 level). We see our results as an indicator that transfer learning can empower RE research by providing a way to train models with less training data and better generalizability at the same time.

As improvements to NoRBERT, we plan to analyze the performance of further language models such as XLNet [39] or XLM [40] and combine NoRBERT with models that perform better on certain tasks. Furthermore, we will investigate how a multiclass multilabel classification approach performs on the classification of functional requirements.

REFERENCES

[1] E. Knauss, S. Houmb, K. Schneider, S. Islam, and J. Jürjens, "Supporting requirements engineers in recognising security issues," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2011, pp. 4–18.

[2] M. Glinz, "On Non-Functional Requirements," in *15th IEEE International Requirements Engineering Conference (RE 2007)*, Oct. 2007, pp. 21–26.

[3] F.-L. Li, J. Horkoff, J. Mylopoulos, R. S. S. Guizzardi, G. Guizzardi, A. Borgida, and L. Liu, "Non-functional requirements as qualities, with a spice of ontology," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, Aug. 2014, pp. 293–302.

[4] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are "non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. Austin, Texas: Association for Computing Machinery, May 2016, pp. 832–842.

[5] 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, May 2007.

[6] I. Hussain, L. Kosseim, and O. Ormandjieva, "Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents," in *Proceedings of the 13th International Conference on Natural Language and Information Systems: Applications of Natural Language to Information Systems*, ser. NLDB '08. London, UK: Springer-Verlag, Jun. 2008, pp. 287–298.

[7] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What Works Better? A Study of Classifying Requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sep. 2017, pp. 496–501.

[8] 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)*, Sep. 2017, pp. 490–495.

[9] 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)*, Sep. 2019, pp. 142–152.

[10] J. Howard and S. Ruder, "Fine-tuned language models for text classification," *CoRR*, vol. abs/1801.06146, 2018. [Online]. Available: http://arxiv.org/abs/1801.06146

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

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

[13] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Supplementary Material of "NoRBERT: Transfer Learning for Requirements Classification"," May 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3833660

[14] J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: A process-oriented approach," *IEEE Transactions on Software Engineering*, vol. 18, no. 6, pp. 483–497, Jun. 1992.

[15] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer Science & Business Media, Dec. 2012.

[16] M. Broy, "Rethinking Nonfunctional Software Requirements," *Computer*, vol. 48, no. 5, pp. 96–99, May 2015.

[17] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects," in *14th IEEE International Requirements Engineering Conference (RE'06)*, Sep. 2006, pp. 39–48.

[18] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, "The promise repository of empirical software engineering data," 2012.

[19] J. Winkler and A. Vogelsang, "Automatic Classification of Requirements Based on Convolutional Neural Networks," in *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, Sep. 2016, pp. 39–45.

[20] R. Navarro-Almanza, R. Juarez-Ramirez, and G. Licea, "Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification," in *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*, Oct. 2017, pp. 116–120.

[21] A. Dekhtyar and V. Fong, "RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, Sep. 2017, pp. 484–489.

[22] S. Amasaki and P. Leelaprute, "The Effects of Vectorization Methods on Non-Functional Requirements Classification," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Aug. 2018, pp. 175–182.

[23] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. Beijing, China: JMLR.org, Jun. 2014, pp. II–1188–II–1196.

[24] D. Mekala, V. Gupta, B. Paranjape, and H. Karnick, "SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 659–669.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Jan. 2013.

[26] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. of NAACL*, 2018.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Long Beach, California, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[30] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 19–27.

[31] I. Tenney, D. Das, and E. Pavlick, "BERT Rediscovers the Classical NLP Pipeline," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4593–4601.

[32] D. P. Kingma and L. J. Ba, "Adam: A Method for Stochastic Optimization," 2015.

[33] I. Loshchilov and F. Hutter, "Fixing Weight Decay Regularization in Adam," Feb. 2018.

[34] X. Liu and A. Wangperawong, "Transfer learning robustness in multi-class categorization by fine-tuning pre-trained contextualized language models," *CoRR*, vol. abs/1909.03564, 2019. [Online]. Available: http://arxiv.org/abs/1909.03564

[35] A. Ghazarian, "Characterization of functional software requirements space: The law of requirements taxonomic growth," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, Sep. 2012, pp. 241–250.

[36] R. Sharma and K. K. Biswas, "Functional requirements categorization Grounded Theory approach," in *2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, Apr. 2015, pp. 301–307.

[37] K. Krippendorff, *Content Analysis: An Introduction to Its Methodology*. SAGE Publications, May 2018.

[38] ——, "Reliability in content analysis: Some common misconceptions and recommendations," *Human communication research*, vol. 30, no. 3, pp. 411–433, 2004.

[39] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 5754–5764.

[40] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale," Nov. 2019.