A Zero-Shot Learning Approach to Classifying Requirements: a Preliminary Study

Waad Alhoshan[1][0000-0002-3433-4653], Liping Zhao[2][0000-0001-8556-8655], Alessio Ferrari[3][0000-0002-0636-5663], and Keletso J. Letsholo[4][0000-0003-4355-7987]

[1] Al-Imam Mohammed ibn Saud Islamic University, Saudi Arabia
wmaboud@imamu.edu.sa
[2] University of Manchester, U.K. liping.zhao@manchester.ac.uk
[3] CNR-ISTI, Pisa, Italy alessio.ferrari@isti.cnr.it
[4] Higher Colleges of Technology, UAE kletsholo@hct.ac.ae

Abstract. *Context and motivation:* Natural Language Processing (NLP) techniques are constantly improving their capabilities, and deep learning approaches are now used in the daily practice of several application domains. Requirements engineering (RE) research has traditionally incorporated NLP solutions to address its fundamental tasks, such as classification, tracing and defect detection. *Question/problem:* However, RE research often suffers from lack of annotated datasets, and this makes it difficult to fully exploit supervised NLP techniques in general, and deep-learning ones in the specific, thereby losing the potential advantages offered by these techniques. *Principal ideas/results:* To address the problem of limited annotated datasets, we propose to use zero-shot classification, and apply this learning paradigm to RE tasks that can be treated as classification problems. We experimented with the task of distinguishing between two types of NFR requirements: usability and security requirement, and obtained encouraging weighted F-scores over 80% and almost perfect recall rates from a number of the tested models, without any training data and fine-tuning. *Contribution:* This work paves the basis for further research in application of zero-shot learning, and towards the solution of the long-standing problem of dataset annotation in RE.

Keywords: Requirements Engineering · Natural Language Processing · Zero-Shot Classification· Machine Learning · Deep Learning · Transfer Learning · Language Models

1 Introduction

Data shortage, particularly lack of annotated task-specific data, has been a major challenge for requirements engineering (RE) researchers interested in applying natural language processing (NLP) and machine learning (ML) techniques to requirements documents [24, 6]. Even for the lively field of app review analysis, Dabrowki *et al.* [3] has shown that most studies have not released their annotated dataset. Also when datasets are available, the annotation process is time consuming and error prone [4], thus calling for solutions that can work well with limited data. Transfer learning makes it possible to address this issue, by training language models on largely available NL datasets, and then fine tuning on a smaller set of domain specific ones [20]. Zero-shot learning further improves the idea by treating sentence classification as a problem of predicting whether a NL sentence is related to a NL tag or not, by reasoning solely on the embedding of the sentence and of the tag, and not resorting on pre-annotated classes for training. In this paper, we perform a feasibility study on using zero-shot learning for the problem of non-functional requirements (NFRs) classification. Our results show comparable performances to other supervised approaches that use a considerable amount of annotated datasets for training or fine-tuning existing models. The affordability of the approach makes it possible to be further investigated and extended. In the paper, we also discuss our future steps in the application of this solution to other classification-related tasks in RE.

2 Background: From Transfer Learning to Zero-Shot Learning

Transfer learning refers to the ability of a ML model to recognize and apply knowledge learned in previous tasks to novel, but related tasks [11]. For example, we can train a model with a sentiment analysis task and then transfer the model to perform a related task such as spam detection [20]. The power of transfer learning lies in enabling a high-performance ML model trained with easily obtained data from one domain to be 'transferred' to a different, but related target domain [11, 20]. In so doing, transfer learning aims to improve the performance of a ML model in the target domain, whilst avoiding much expensive data-labeling efforts and alleviating the training data shortage problem [11, 20, 17].

Transfer learning has become a commonplace in advanced *language models (LMs)*, as these models can be *pre-trained* on a data-rich task before being *fine-tuned* on different downstream tasks [14]. In particular, the LMs such as BERT [5] and GPT [13] allow a model to be pre-trained using *unsupervised learning* on *unlabeled data* and then fine-tuned using *supervised learning* on *labelled data* from the downstream tasks. These approaches have achieved state-of-the-art results in many of the most common NLP benchmark tasks [5, 14].

What makes BERT and GPT so powerful is their underlying *Transformer* architecture [18]. The Transformer architecture modifies the Recurrent Neural Network (RNN) by replacing its recurrent layers with multi-headed *self-attention* functions, which transform an input sequence of tokens into a vector representation of weighted values [18]. However, while Transformer-based LMs can avoid expensive data-labeling efforts, they are very expensive to pre-train, as they require a large amount of training data, as well as expensive computational resources. For example, pre-training of BERT-base requires 128,000 words $\times$ 1,000,000 steps on 4 Cloud TPUs with 16 TPU chips for four days [5]. Although fine-tuning of these models is relatively less expensive, it still requires thousands or tens of thousands of labelled task-specific examples [5, 2].

Zero-shot learning (ZSL) has been originally used in image processing to predict unseen images [16], and has recently been adapted to text classification to predict unseen classes [12]. Unlike other text classifiers that learn to classify a given sentence as one of the possible classes, the ZSL models (also called *learners*) learn to predict *whether the sentence is related to a tag or not*. Thus, ZSL treats a classification task (binary, or multi-class) as a problem of finding relatedness between sentences and classes [12]. To train a ZSL model, we need to add all the tags (labels) to each sentence in the training set for the model to learn the likelihood of each tag for each sentence. Two general methods are available for training a ZSL model: the *embedding-based* method that integrates the text embedding layer with the tag embedding layer, and then measures the probability of their relatedness using some similarity function [12]; the *entailment-based* method that treats an input text sequence as a premise and the candidate tags as a hypothesis, and then infers if the input text is an entailment of any of the given tags or not [23]. The real potential of ZSL is its partnership with large pre-trained LMs. By piggybacking on such models, the ZSL models can perform competitively on downstream NLP tasks *without fine-tuning (zero-shot) or with only a handful of labelled task-specific examples (few-shot)*, thus removing the burden of expensive data-labeling efforts, the goal set out by transfer learning.

3 Preliminary Study

Dataset In this study we use the popular PROMISE dataset [5] to demonstrate the potential of ZSL for requirements classification. The dataset contains 625 FRs and NFRs with associated labels, and has frequently been used as a benchmark for requirements classification [8, 4, 7]. In our feasibility study, we select only the subset of the dataset that contains the usability and security requirements. The two classes of requirements are evenly distributed, with 67 usability and 66 security requirements, labelled as US and SE respectively. The classification task in our study is to apply different ZSL models to predict *whether a requirement in this dataset is related to a usability tag or not, or whether a requirement in this dataset is related to a security tag or not.*

Setting-up the ZSL Classifiers We use the embedding-based method to study the ZSL models and select the following nine pre-trained Transformer models from Hugging Face models hub [21]:

- **BERT family** [5]: $\text{BERT}_{\text{base-uncased}}$, $\text{BERT}_{\text{base-cased}}$, $\text{BERT}_{\text{large-uncased}}$, $\text{BERT}_{\text{large-cased}}$;
- **RoBERTa family** [9]: $\text{RoBERT}_{\text{base}}$, $\text{XLM-RoBERT}_{\text{base}}$;
- **XLNet family** [22]: $\text{XLNet}_{\text{base-cased}}$;
- **Sentence-based LMs** : Sentence-BERT [15] and MiniLM-L12-v2 [19] which is fine-tuned by one billion sentence pairs dataset from different online technical feeds such as Reddit comments.

Based on these nine LMs, we build nine embedding-based ZSL classifiers by fitting each model to the default ZSL pipeline from the Transformers library [21]. Afterwards we apply each ZSL classifier as follows: 1) We feed each requirement sentence and its labels to the classifier. 2) The LM within the classifier carries out tokenization and then creates a sentence embedding and a label embedding layer. 3) The classifier processes the embedding results by computing the relatedness between the sequences embedding and the label embeddings using cosine similarity. 4) Finally, the overall similarity scores are fed into a classification function, and the probabilities of all labels are computed to select the maximum score as the most related label to a given requirement.

[5] https://doi.org/10.5281/zenodo.268542

Table 1. The Experiment Results, the **bold** font indicates the best results obtained from the ZSL-based experiments, and the underlined scores refer to the best results reported in the related work [8] and [7].

Classification Approach	**Usability (US)**			**Security (SE)**			**A**	**w.F**
	P	**R**	**F1**	**P**	**R**	**F1**		
[8] Supervised$_{\text{mulit}}$ (w/o feature sel.)	0.65	0.82	0.70	0.81	0.77	0.74	0.76	0.73
[8] Supervised$_{\text{mulit}}$ (500 best features)	0.70	0.66	0.64	0.64	0.53	0.56	0.61	0.6
[7] NoRBERT$_{\text{base multi}}$ (ep.=50)	0.78	0.85	0.81	0.78	0.92	0.85	0.85	0.83
[7] NoRBERT$_{\text{large multi}}$ (ep.=50)	0.83	0.88	0.86	0.90	0.92	0.91	0.87	0.86
ZSL BERT$_{\text{base-uncased}}$	**0.83**	0.52	0.64	0.65	0.90	0.75	0.71	0.70
ZSL BERT$_{\text{base-cased}}$	**0.83**	0.58	0.68	0.68	0.88	0.77	0.73	0.73
ZSL BERT$_{\text{large-uncased}}$	**0.83**	0.15	0.26	0.54	0.97	0.69	0.56	0.48
ZSL BERT$_{\text{large-cased}}$	0.52	0.18	0.27	0.51	0.84	0.63	0.51	0.45
ZSL RoBERTa$_{\text{base}}$	0.00	0.00	0.00	0.50	**1.00**	0.67	0.50	0.34
ZSL XLM-RoBERTa$_{\text{base}}$	0.49	**1.00**	0.66	0.00	0.00	0.00	0.50	0.33
ZSL XLNet$_{\text{base-cased}}$	0.47	0.68	0.56	0.45	0.25	0.32	0.47	0.44
ZSL Sentence BERT	0.71	0.80	0.76	0.78	0.69	0.73	0.74	0.75
ZSL MiniLM-L12-v2	0.73	**1.00**	**0.85**	**1.00**	0.64	**0.78**	**0.82**	**0.82**

Evaluation and Results Analysis We implement the ZSL classifiers on Google Colab with a standard CPU at 2.30GHz with 12GB of RAM. The entire experiment took less than 5 minutes (4.39 mis) to run, with 0.77GB of RAM usage. The results are then exported into structured (.csv) files for further investigation [6]. We computed the ZSL classifiers performance in comparison to the original annotated PROMISE dataset. For performance evaluation, we use precision (P), recall (R), F1, weighted F-score (w.F), and accuracy (A). Results are shown in Table 1.

Considering that our ZSL classification models have not been trained on any sample requirements from the dataset, in contrast to fully supervised or fine-tuned classification approaches, the reported results from some of the used LMs are considered to be *encouraging* for further investigation. In particular, we notice that recall (R) is equal to 100% for some of the LM, as recommended by Berry [1]. We compared the performance of ZSL classifiers with existing work ([8] and [7]) which used the same NFR dataset. The results provided by fine-tuned BERT$_{\text{large}}$ model namely NoRBERT [7] has still the highest performance rates in terms of precision rates and F1 scores. However, one of the ZSL classifier, which applied Sentence-based (MiniLM-L12-v2 LM [19], has a comparable performance of a weighted F-score of 82% comparing to 86% provided by NoRBERT$_{\text{large}}$ model. In addition, two of the ZSL classifier models which are based on sentence embeddings (Sentence BERT and MiniLM-L12-v2) have outperformed the fully-supervised learning approaches by Kurtanovic and Maalej [8] with more than 75% of weighted F-scores. Example requirements with their similarity scores according to the given labels set and based on Sentence-based embedding are shown in Table 2.

Overall, the sentence-based LM with ZSL classifier have provided almost best results in our initial experiments comparing to other word-based LM (e.g., BERT, RoBERTa, and XLNet). This observation suggests that sentence-based LMs are to be preferred over word-based LMs as methods for generating requirement and label embedding. Sentence-based LMs are different from word-based LMs. A word-based LM aims to learn the probability of words in entire dataset, while a sentence-based LM aims to contextualize the words at sentence-level, thus exploiting the whole sentence structure and

[6] The results are available at: https://github.com/waadalhoshan/ZSL4REQ

Table 2. Requirement examples with their associated similarity scores from sentence-based embedding with ZSL. The strike(*) refers to a mislabeling.

Requirement Text	***Label***	**Sentence-BERT**	**Sentence-Transformers**
The website will provide a help section with answers and solutions to common problems.	*US*	Usability: 0.26	Usability: 0.14
Data integrity scripts will be run on a weekly basis to verify the integrity of the database.	*SE*	Security: 0.18	Security: 0.20
The product should be able to be used b 90% of novice users on the Internet.	*US*	*Security: 0.17	Usability: 0.25
The product shall conform to the Americans with Disabilities Act.	*US*	Usability: 0.37	*Security: 0.19

content in the learning process. As a next step of this research, we will fine-tune an existing sentence-based LM (e.g., Sentence BERT) for specific RE tasks. In the following section, we will briefly outline our future research plan.

4 Conclusion and Future Plan

The promising performance of ZSL observed in our study indicates its potential for requirements classification. We plan to expand this study to conduct further experiments. First, we will extend the current approach to the entire PROMISE dataset, to consider more fine-grained semantic categories, similar to the work of Hey *et al.* [7]. To this end, we plan to experiment different deep learning architectures for implementing the ZSL requirements classifier, as described by Pushp and Srivastava [12], and and then apply different fine-tuning techniques to the LMs with promising performances, such as sentence-BERT model [15]. For example, by training the high-level layer of the pretrained LM and freeze the low-level layers, and then fine-tuning by freezing the entire LM and train additional layers on the top. In addition, we will expand the fine-tuning and training of the LMs to different requirement datasets and different classification taxonomies. Second, we will extend the ZSL approach to the few-shot learning (FSL) by using one shot (one labelled task-specific requirement) and a few shot (a handful of labelled requirements) to fine-tune the LMs to see if the performance of ZSL can be improved. According to the study carried out by OpenAI [2], even using just one labelled example can substantially improve the classifier performance.

Finally, we will apply ZSL/FSL to other classification related tasks in RE such as those identified in our recent mapping study [24], including *Detection* (detecting linguistic issues in requirements documents) and *Extraction* (identifying key domain abstractions and concepts). We will also repeat our study on the app review classification problem, addressed, e.g., by [10], to see if the models we developed for requirements classification can be transferred to the app review classification task.

References

1. Berry, D.M.: Empirical evaluation of tools for hairy requirements engineering tasks. Empirical Software Engineering **26**(6), 1–77 (2021)

2. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
3. Dabrowski, J., Letier, E., Perini, A., Susi, A.: App review analysis for software engineering: A systematic literature review. University College London, Tech. Rep (2020)
4. Dalpiaz, F., Dell'Anna, D., Aydemir, F.B., Çevikol, S.: Requirements classification with interpretable machine learning and dependency parsing. In: RE'19. pp. 142–152. IEEE (2019)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Ferrari, A., Dell'Orletta, F., Esuli, A., Gervasi, V., Gnesi, S.: Natural language requirements processing: A 4d vision. IEEE Softw. **34**(6), 28–35 (2017)
7. Hey, T., Keim, J., Koziolek, A., Tichy, W.F.: NoRBERT: Transfer learning for requirements classification. In: RE'20. pp. 169–179. IEEE (2020)
8. Kurtanović, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: RE'17. pp. 490–495. Ieee (2017)
9. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
10. Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the automatic classification of app reviews. REJ **21**(3), 311–331 (2016)
11. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2009)
12. Pushp, P.K., Srivastava, M.M.: Train once, test anywhere: Zero-shot learning for text classification. arXiv preprint arXiv:1712.05972 (2017)
13. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. Technical Report, OpenAI (2018)
14. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019)
15. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019)
16. Romera-Paredes, B., Torr, P.: An embarrassingly simple approach to zero-shot learning. In: ICML'15. pp. 2152–2161 (2015)
17. Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T.: Transfer learning in natural language processing. In: NACL'19. pp. 15–18 (2019)
18. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS'17. pp. 5998–6008 (2017)
19. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. arXiv preprint arXiv:2002.10957 (2020)
20. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big data **3**(1), 1–40 (2016)
21. Wolf, T., Debut, L., Sanh, V., et al.: Transformers: State-of-the-art natural language processing. In: EMNLP'20. pp. 38–45 (2020)
22. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. NeurIPS'19 **32** (2019)
23. Yin, W., Hay, J., Roth, D.: Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. CoRR **abs/1909.00161** (2019)
24. Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K.J., Ajagbe, M.A., Chioasca, E., Batista-Navarro, R.T.: Natural language processing for requirements engineering: A systematic mapping study. ACM Comput. Surv. **54**(3), 55:1–55:41 (2021)