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

Waad Alhoshan[1](✉), Liping Zhao[2](✉), Alessio Ferrari[3],
and Keletso J. Letsholo[4]

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

Abstract. *Context and motivation:* Advances in Machine Learning (ML) and Deep Learning (DL) technologies have transformed the field of Natural Language Processing (NLP), making NLP more practical and accessible. Motivated by these exciting developments, Requirements Engineering (RE) researchers have been experimenting ML/DL based approaches for a range of RE tasks, such as requirements classification, requirements tracing, ambiguity detection, and modelling. *Question/problem:* Most of today's ML/DL approaches are based on supervised learning techniques, meaning that they need to be trained using annotated datasets to learn how to assign a class label to examples from an application domain. This requirement poses an enormous challenge to RE researchers, as the lack of requirements datasets in general and annotated datasets in particular, makes it difficult for them to fully exploit the benefit of the advanced ML/DL technologies. *Principal ideas/results:* To address this challenge, this paper proposes a novel approach that employs the Zero-Shot Learning (ZSL) technique to perform requirements classification. We build several classification models using ZSL. We focus on the classification task because many RE tasks can be solved as classification problems by a large number of available ML/DL methods. In this preliminary study, we demonstrate our approach by classifying non-functional requirements (NFRs) into two categories: Usability and Security. ZSL supports learning without domain-specific training data, thus solving the lack of annotated datasets typical of RE. The study shows that our approach achieves an average of 82% recall and F-score. *Contribution:* This study demonstrates the potential of ZSL for requirements classification. The promising results of this study pave the way for further investigations and large-scale studies. An important implication is that it is possible to have very little or no training data to perform requirements classification. The proposed approach thus contributes to the solution of the long-standing problem of data shortage in RE.

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

V. Gervasi and A. Vogelsang (Eds.): REFSQ 2022, LNCS 13216, pp. 52–59, 2022.
https://doi.org/10.1007/978-3-030-98464-9_5

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 [6,24]. 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 (ZSL) 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 preliminary 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 many expensive data-labeling efforts and alleviating the training data shortage problem [11,17,20].

Transfer learning has become a commonplace in advanced *language models (LMs)*, which are machine learning frameworks for NLP tasks. 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], which transforms a given sequence of elements, such as the sequence of words in a sentence, into another sequence, with the help of an Encoder and a Decoder. The output sequence can be in another language, symbols, a copy of the input, etc. Both Encoder and Decoder are composed of modules, which are made of multi-head *self-attention* functions and feed forward layers. The self-attention mechanism looks at an input sequence and decides at each step which parts of the sequence are important or unimportant; which parts should be remembered or forgotten. As sentences in natural language are sequence-dependent—that is, the order of the words is crucial for understanding a sentence—, transformers are particularly useful for NLP tasks, such as language understanding and machine translation. In addition, transformers are capable of performing transformation sequentially as well as in parallel, by stacking multiple self-attention mechanisms on top of each other or by using them side by side. 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, to pre-train a BERT$_{\text{base}}$ model[1], it requires 128,000 words × 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 [2,5].

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 classifier, 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. The learning involves measuring the semantic relatedness of a given input sequence (e.g., a sentence) to each class or tag and assigning a probabilistic score to the input in relation to the tag to establish if the input belongs to the corresponding class. After the assignment of all the tags to the input, the classifier proposes a threshold value to suggest if the input should be classified into one or more classes represented by the tags. Effectively, ZSL performs a multi-label classification for each input.

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. There are two general methods which are available for training a ZSL model: the *embedding-based* method and the *entailment-based* method. The *embedding-based* method integrates the text embedding layer with the tag embedding layer, and then measures the probability of their relatedness using some similarity function [12]. The embeddings are commonly extracted from pre-trained LMs and the embedding layers could be at the word or sentence level, a word-based embedding layer aims to learn the probability of words in entire datasets, while a sentence-based embedding layer aims to contextualize the words at sentence-level, thus exploiting the

[1] BERT has two basic models: BERT$_{\text{base}}$ and BERT$_{\text{large}}$. BERT$_{\text{base}}$ has 12 encoder layers whereas BERT$_{\text{large}}$ has 24. Which BERT model to use depends on the application and BERT$_{\text{base}}$ is usually sufficient for experiments, as it takes less time and resource to fine-tune comparing to BERT$_{\text{large}}$.

whole sentence structure and content in the learning process. On the other hand, the *entailment-based* method 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]. In this preliminary study, we choose the embedding-based method due to the widely availability of embedding technologies.

3 Preliminary Study

Dataset In this study we use the popular PROMISE dataset[2] 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 [4,7,8]. 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 implement 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.

Evaluation and Results Analysis. We implement the ZSL classifiers on Google Colab with a standard CPU at 2.30 GHz with 12 GB of RAM. The entire experiment took less than 5 min (4.39 mis) to run, with 0.77 GB of RAM usage. The results are then exported into structured (.csv) files for further investigation[3]. 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.

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

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

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$_{\mathrm{mulit}}$ (w/o feature sel.)	0.65	0.82	0.70	0.81	0.77	0.74	0.76	0.73
[8] Supervised$_{\mathrm{mulit}}$ (500 best features)	0.70	0.66	0.64	0.64	0.53	0.56	0.61	0.6
[7] NoRBERT$_{\mathrm{basemulti}}$ (ep. = 50)	0.78	0.85	0.81	0.78	0.92	0.85	0.85	0.83
[7] NoRBERT$_{\mathrm{largemulti}}$ (ep. = 50)	0.83	0.88	0.86	0.90	0.92	0.91	0.87	0.86
ZSL BERT$_{\mathrm{base\text{-}uncased}}$	**0.83**	0.52	0.64	0.65	0.90	0.75	0.71	0.70
ZSL BERT$_{\mathrm{base\text{-}cased}}$	**0.83**	0.58	0.68	0.68	0.88	0.77	0.73	0.73
ZSL BERT$_{\mathrm{large\text{-}uncased}}$	**0.83**	0.15	0.26	0.54	0.97	0.69	0.56	0.48
ZSL BERT$_{\mathrm{large\text{-}cased}}$	0.52	0.18	0.27	0.51	0.84	0.63	0.51	0.45
ZSL RoBERTa$_{\mathrm{base}}$	0.00	0.00	0.00	0.50	**1.00**	0.67	0.50	0.34
ZSL XLM-RoBERTa$_{\mathrm{base}}$	0.49	**1.00**	0.66	0.00	0.00	0.00	0.50	0.33
ZSL XLNet$_{\mathrm{base\text{-}cased}}$	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**

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$_{\mathrm{large}}$ model namely NoRBERT [7] has still the highest performance rates in terms of precision rates and F1 scores. However, one of the ZSL classifiers, 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$_{\mathrm{large}}$ 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. However, we noticed that word-based LMs can be biased towards a specific label. For example, both RoBERTa models (i.e., RoBERTa$_{\mathrm{base}}$ and XLM-RoBERTa) are word-based and have the tendency to label all the requirements as Security only or Usability only, as shown in the recall and F1-score results in Table 1. This is predictable with any pre-trained LMs as those models are trained on general-domain datasets, making them less accurate when working with domain-specific data. Therefore, what we regard as a Security requirements (as requirement engineers) could be classified by those general models into more general categories not related to the security aspects of a software.

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. 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.

Table 2. Requirement examples with their associated similarity scores using the ZSL classifier which is based on the sentence-based embedding. 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

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 then apply different fine-tuning techniques to the LMs with promising performances, such as the Sentence-BERT model [15]. For example, by training the high-level layer of the pre-trained 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 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. Empir. Softw. Eng. **26**(6), 1–77 (2021). https://doi.org/10.1007/s10664-021-09986-0
2. Brown, T.B., 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, Technical report (2020)
4. Dalpiaz, F., Dell'Anna, D., Aydemir, F.B., Çevikol, S.: Requirements classification with interpretable machine learning and dependency parsing. In: RE 2019, 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 2020, pp. 169–179. IEEE (2020)
8. Kurtanović, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: RE 2017, pp. 490–495. IEEE (2017)
9. Liu, Y., et al.: 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. Requirements Eng. **21**(3), 311–331 (2016). https://doi.org/10.1007/s00766-016-0251-9
11. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **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., et al.: 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 2015, pp. 2152–2161 (2015)
17. Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T.: Transfer learning in natural language processing. In: NACL 2019, pp. 15–18 (2019)
18. Vaswani, A., et al.: Attention is all you need. In: NeurIPS 2017, 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.D.: A survey of transfer learning. J. Big Data **3**(1), 1–40 (2016). https://doi.org/10.1186/s40537-016-0043-6

21. Wolf, T., Debut, L., Sanh, V., et al.: Transformers: state-of-the-art natural language processing. In: EMNLP 2020, 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. In: NeurIPS 2019, vol. 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., et al.: Natural language processing for requirements engineering: a systematic mapping study. ACM Comput. Surv. **54**(3), 55:1–55:41 (2021)