Automated Requirements Demarcation using Large Language Models: An Empirical Study

Kaishuo Wang, Feier Zhang and Mehrdad Sabetzadeh

School of Electrical Engineering and Computer Science (EECS), University of Ottawa, 800 King Edward Ave, Ottawa, ON, K1N 6N5, Canada

Abstract

Requirements demarcation is concerned with the identification of software and systems requirements in technical natural-language specifications. Manually distinguishing requirements from non-requirements is a labour-intensive task and is prone to omissions and errors. Therefore, building a reliable and accurate automated method for requirements demarcation is important. This research evaluates auto-encoder large language models [1], auto-regressive large language models [2], few-shot learning, and ensembling methods for accurate demarcation of requirements. Specific architectures considered are DeBERTa, Llama2, and few-shot learning with RoBERTa. Our work empirically compares these approaches and determines which one yields the best accuracy. Our experimental results show that DeBERTa yields the best performance. While Llama2 requires more computational resources and training time compared to DeBERTa, it has an accuracy deficit of approximately 1% across different metrics. This result suggests that auto-encoder models may be more suitable for requirements demarcation than auto-regressive models. Further, we observe that the few-shot learning approach has the worst performance among the alternatives considered. Finally, we find that ensembling leads to minor performance improvements compared to a single model. We make all the artifacts developed as part of this research available online: *https://github.com/KaishuoWang/Automated-Requirements-Classification*.

Keywords

requirements demarcation, natural language processing, ensemble learning, transformers, empirical evaluation

1. Introduction

Distinguishing requirements from non-requirements in technical documents is a difficult and labour-intensive task [3]. Manually classifying text across thousands of pages to identify requirements is also prone to omissions and errors. Automating requirements demarcation, defined as the task of distinguishing requirements from non-requirement sentences [4], is thus important for streamlining downstream analysis, traceability, risk identification, and testing [3]. Table 1 shows examples to illustrate sentences classified as requirements and non-requirements.

With the recent emergence of large language models, an interesting question arises as to whether one can improve upon the accuracy of existing natural language processing techniques

In: D. Mendez, A. Moreira, J. Horkoff, T. Weyer, M. Daneva, M. Unterkalmsteiner, S. Bühne, J. Hehn, B. Penzenstadler, N. Condori-Fernández, O. Dieste, R. Guizzardi, K. M. Habibullah, A. Perini, A. Susi, S. Abualhaija, C. Arora, D. Dell'Anna, A. Ferrari, S. Ghanavati, F. Dalpiaz, J. Steghöfer, A. Rachmann, J. Gulden, A. Müller, M. Beck, D. Birkmeier, A. Herrmann, P. Mennig, K. Schneider. Joint Proceedings of REFSQ-2024 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2024. Winterthur, Switzerland, April 8, 2024.

kwang126@uottawa.ca (K. Wang); fzhan081@uottawa.ca (F. Zhang); msabetza@uottawa.ca (M. Sabetzadeh)

CEUR Workshop Proceedings (CEUR-WS.org)

Table 1
Examples of Requirements and Non-requirements

Text	Label
Upon installation, a DigitalHome user account shall be established.	Requirement
NPAC SMS shall provide post-collection audit analysis tools that can produce detailed reports on data items relating to system intrusions.	Requirement
NANC Version 1.6, released on 11/12/97, contains changes from the NANC FRS Version 1.5.	Non-requirement
User selects option to associate file types with editors (ALT 1).	Non-requirement

for requirements demarcation. This paper reports on an empirical examination comparing several alternative technologies. We evaluate two state-of-the-art models, DeBERTa and Llama2, as well as few-shot learning with RoBERTa. We also examine an ensemble approach combining these methods to study whether demarcation accuracy over complex documents can be further improved. Based on our experimental results, we observe that DeBERTa slightly outperforms our replication of Bashir et al.'s approach [3]. The accuracy results reported by Bashir et al. are nonetheless slightly higher than our replication, most likely due to slight differences in hyperparameter optimization. Consequently, our accuracy results from DeBERTa are slightly below those reported by Bashir et al. In view of this finding, we hypothesize that DeBERTa will likely outperform Bashir et al.'s approach if its hyperparameters are optimized according to the process used by Bashir et al., details of which we did not have and could not exactly replicate. As for Llama2, we observe that the model has lower performance than DeBERTa, which could be attributed to the small size of the training data, the lack of context, and the absence of a training data selection strategy. We further observe that few-shot learning has a notable performance deficit compared to both DeBERTa and Llama2. In addition, we find that simply integrating the predictions of each approach using their normalized F1 score does not lead to significant improvement in performance. More complex ensemble approaches should thus be considered in the future.

2. Background and Related Work

Automating the demarcation of requirements has seen considerable interest in the field of software engineering. This section presents a brief overview of requirement demarcation techniques alongside their enabling techniques, highlighting the evolution from traditional machine learning approaches to the latest advancements in deep learning and transformer-based models.

Early requirements demarcation methods, e.g., work by Abualhaija et al. [4], use traditional machine learning such as Support Vector Machines (SVM), Logistic Regression (LR), and Naive Bayes (NB). These methods require careful feature engineering and show limitations in handling the linguistic nuances and complexities inherent in requirements text.

Long Short-Term Memory (LSTM) [5] networks revolutionized deep learning for sequential data. LSTMs have gating mechanisms that allow them to retain or forget information, reducing issues like vanishing gradients. This enables models to process longer sequences while capturing context. New word embeddings like FastText (FT) [6] and Global Vectors (GloVe) [7] have been used as well, with FT aggregating subword n-grams for better morphology understanding and GloVe using co-occurrence statistics to capture semantic relationships. Overall, LSTMs allow sequential modelling of longer texts, while new word embeddings like FT and GloVe enable more semantic understanding. In addition, Winkler et al.[8] proposed a method that utilizes convolutional neural networks for automated requirements classification. Their method achieved 0.73 in accuracy and 0.89 in recall on a real-world automotive requirements specification

The emergence of Transformer-based models marks an important milestone in NLP. Transformers like BERT [1] have brought about major advances in NLP through self-attention and bidirectional context modelling. Rather than processing text linearly, self-attention weighs the significance of each word relative to all others, learning context more effectively. BERT specifically introduces bidirectionality, understanding context based on surrounding text on both sides of a word. This enables capturing nuances and relationships that were previously difficult to capture. BERT uses WordPiece tokenization [9] to break words into subword units, allowing it to handle unseen words by representing them as known subwords. Therefore, BERT and its variants generally perform well at understanding context and complex dependencies, which are key attributes for accurately demarcating the content of complex requirements documents.

Bashir et al. [3] propose few-shot learning with sentence transformers [10], utilizing the SetFit framework [11] to fine-tune various pre-trained Sentence Transformer models for the task of demarcating requirements. This process involves a dual-step training approach: initially, fine-tuning the Sentence Transformer model on a limited dataset using a contrastive training method, followed by training a Logistic Regression model to act as a classification head on the embeddings generated by the fine-tuned Sentence Transformer. The evaluation of this model entails generating sentence embeddings from unseen examples and then predicting the class label with the Logistic Regression model.

In Bashir et al.'s work, the *bert-base-uncased* pipeline obtained the best performance, with a Macro average F1 score of 0.83 on the Dronology dataset [12]. This pipeline will be used as our baseline and compared to our approaches.

3. Alternative Approaches

Figure 1 provides an overview of the alternative models we examine in this paper. Recognizing the potential for synergy, we consider two ensemble systems among the alternatives. The first ensemble system integrates the strengths of DeBERTa and Llama2, while the second system leverages the strength of all three models. This approach not only allows us to compare three methods separately but also to investigate potential enhancements brought about by ensembling.

3.1. DeBERTa

DeBERTa [13] improves upon the BERT architecture in several key ways. DeBERTa's defining feature is its *disentangled attention* mechanism. Unlike BERT, which processes content and position information jointly within a single self-attention mechanism, DeBERTa separates the

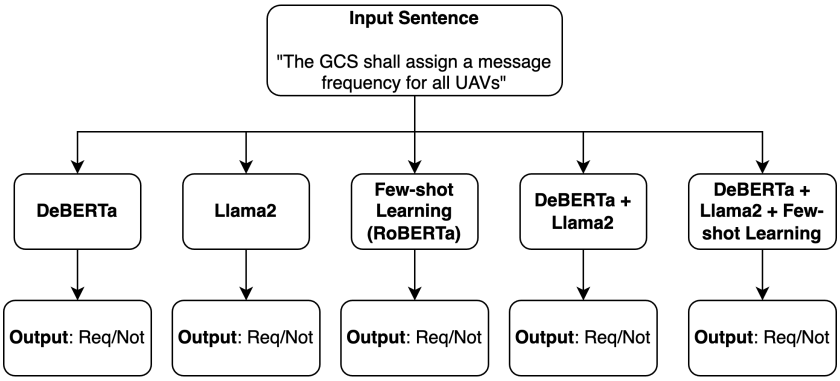
Input Sentence
"The GCS shall assign a message frequency for all UAVs"
DeBERTa
Llama2
Few-shot Learning (RoBERTa)
DeBERTa + Llama2
DeBERTa + Llama2 + Few-shot Learning
Output: Req/Not
Output: Req/Not
Output: Req/Not
Output: Req/Not
Output: Req/Not

Figure 1: Techniques Explored in This Paper for Requirements Demarcation.

two. This allows the model to learn the representation of word positions and the content more effectively, giving it a more refined understanding of the meaning that word order and position contribute to a sentence. Such a capability is especially critical in requirement demarcation tasks, where the position of terms can dramatically change their meaning, and then change the classification. In addition, DeBERTa introduces an *Enhanced Mask Decoder* (EMD). BERT adds absolute position encodings directly in the input layer. In contrast, DeBERTa captures only relative position information within the Transformer layers, and applies absolute position information later, right before predicting masked tokens. Thus, EMD is better at predicting the original tokens from masked ones by using contextual information more efficiently. Given these architectural improvements and DeBERTa's demonstrated superior performance over BERT in various natural language processing benchmarks, including text classification, it is interesting to examine the application of DeBERTa as a potentially more effective alternative to BERT.

3.2. Llama2

Llama2 [2] is a transformer-based auto-regressive large language model. This type of model is often applied to text generation tasks, such as machine translation, generative question-answering systems, and virtual assistants. In this project, we aim to examine the performance of this model on a text classification task (i.e., requirements demarcation) and compare it with BERT and other machine learning-based classification methods. Llama2 has a substantially larger number of parameters than BERT variants. The version we used in this project contains 7 billion parameters, compared to 345 million and 110 million parameters in DeBERTa-large and BERT-base, respectively. With more parameters, Llama2 is better at generalizing knowledge learned from large amounts of training data to new tasks. At the same time, more parameters may improve context understanding and robustness to ambiguity. These improvements will give models a deeper understanding of context and remove ambiguities common in natural language. In addition, Llama2 combines multi-task learning to jointly train models on different natural language understanding tasks. In other words, the Llama2 model is not only trained on a large text corpus but is also fine-tuned for multiple natural language understanding tasks. This enables the model to learn from diverse and rich data sources and improve its generalization capabilities. These improvements by Llama2 present an opportunity for a potentially more accurate and streamlined approach to classification tasks.

3.3. Few-shot Learning

As a machine learning technology that has emerged in recent years, few-shot learning enables the model to learn from a small number of samples. Compared to supervised machine learning, few-shot learning only requires a small number of data points to learn the information in the data and generalize it to new tasks, making it useful when the amount of labelled data is small. Since engineering specification documents are usually confidential information within a company or organization, we hypothesize that applying few-shot learning to the task of requirements demarcation can solve the problem of small data volumes.

One recent development in the field of few-shot learning is SetFit [11] – an efficient and prompt-free framework for the few-shot fine-tuning of sentence-transform models. Compared to other few-shot learning methods, SetFit requires no prompts at all and can generate rich embeddings directly from text examples. In addition, SetFit does not require large models like GPT or Llama to achieve high accuracy. This means that it requires less computing resources and training time.

3.4. Ensemble System

Ensemble learning is a machine learning technique that combines the predictions from two or more models. Compared to using single models, the ensemble technique exploits the predictive power of single models to achieve more accurate predictions and better performance. Moreover, the ensemble technique also improves the robustness of the model by reducing the spread or dispersion of the predictions and model performance.

To ensemble the predictions of DeBERTa, Llama2, and Few-shot Learning, we use the normalized macro F1 score of each model as weight and multiply with their predictions to get the final prediction P:

$$P = w_1 \times P_1 + w_2 \times P_2 + w_3 \times P_3 \tag{1}$$

where w_1, w_2, w_3 are the weights of each model and P_1, P_2, P_3 are the predictions of each model.

4. Implementation

For this project, we utilize the PyTorch and HuggingFace transformers library. During the fine-tuning process of DeBERTa, we add a classification header on top of DeBERTa. This is a common way of fine-tuning, where the weights of a neural network in the classification head are updated via back propagation. However, this method requires a lot of computing resources and time, so it is not suitable for fine-tuning the more complex Llama2 model. Therefore, we use Low Rank Adaptation (LoRA) [14], an improved fine-tuning method that has been proven effective, in the process of fine-tuning Llama2. LoRA avoids catastrophic forgetting, a phenomenon that occurs when knowledge of a pre-trained model is lost during fine-tuning, by fine-tuning two smaller matrices that approximate the weight matrix of a large pre-trained language model. This method greatly reduces the number of trainable parameters, enabling us to keep the computational resources and time required for fine-tuning within an acceptable range. For few-shot learning, we use the SetFit [11] framework and RoBERTa-large model. We fine-tuned the few-shot learning model with different numbers of labelled examples for each label, including 8, 16, 24, and 32. The model fine-tuned with 24 labelled datapoints (per class) yielded the best performance. Therefore, we use 24 randomly selected samples from each label (requirement or non-requirement) for fine-tuning the few-shot learning model.

We fine-tuned the DeBERTa and few-shot learning models using a single Nvidia V100 16GB GPU, and the Llama2 model was fine-tuned using a single Nvidia A100 40GB GPU on Colab.

5. Empirical Evaluation

5.1. Research Questions

The goals of our evaluation are three-fold. First, we aim to investigate whether there is sufficient rationale for replacing auto-encoder models such as BERT and DeBERTa with more complex auto-regressive models like Llama2 for the task of requirements demarcation. Settling this question necessitates an examination of the trade-off between the potentially increased accuracy from the more complex models and the less resource-intensive nature of earlier auto-encoder models. In this comparison, the BERT model was used as baseline to compare with DeBERTa and Llama2, since the BERT model yielded the best performance in [3]. Second, we seek to study the performance of few-shot learning for requirements demarcation to conduct a more thorough comparison with DeBERTa and Llama2. And third, in order to enhance the reliability of the model as a replacement for manual work, we aim to analyze whether the accuracy of requirements demarcation can be further improved by utilizing ensemble learning techniques. To achieve these objectives, we present the following research questions (RQs):

RQ1: Which model architecture - DeBERTa, Llama or BERT - yields the most accurate requirements demarcation results?

RQ2: Can few-shot learning achieve on-par performance with state-of-the-art approaches?

RQ3: Can ensembling improve upon individual models?

To answer the research questions, we designed two sets of experiments. In the first set of experiments, we utilize stratified five-fold cross-validation on the Dronology dataset (discussed in Section 5.2) with DeBERTa, Llama2, and a RoBERTa-based few-shot learning model to provide a fair and direct comparison with the work of Bashir et al [3]. In the second set of experiments, we fine-tune DeBERTa, Llama2, and the same RoBERTa-based few-shot learning model with the dataset combining the Dronology and PURE datasets (discussed in Section 5.2) and compared their performance.

5.2. Description of Dataset

The dataset we use for our evaluation combines two existing labelled datasets: the Dronology dataset from [12] and a manually extracted and labelled dataset from the PURE dataset by

Table 2
Experimental Results for Accuracy and Training Time on the Dronology Dataset

Model	Accuracy	Weighted Avg.			Macro Avg.			Training Time
		Precision	*Recall*	*F1*	*Precision*	*Recall*	*F1*	
BERT[*]	0.8487	0.8680	0.8487	0.8527	0.8065	0.8418	0.8161	0:11:29
DeBERTa	**0.8701**	0.8703	**0.8701**	**0.8689**	**0.8410**	0.8205	**0.8287**	1:07:08
Llama2	0.7108	0.6941	0.7108	0.7002	0.6099	0.5994	0.6041	1:08:17
Few-shot Learning	0.6794	0.7828	0.6794	0.6955	0.6736	0.7133	0.6520	0:26:41

[*]Results from our replication of *bert-base-uncased* on the Dronology dataset using the same hyper-parameter reported in [3].

Ivanov et al. [15].

The Dronology dataset consists of 398 entries of various classes, including *components*, *design definitions*, *sub-task*, and *requirements*. This dataset was processed and labelled by Bashir et al. [3]. They first labelled all non-requirement classes as *non-requirement*, then deleted 19 entries without text. After processing, the dataset contains 99 requirements and 280 non-requirements. To mitigate imbalance, the authors use stratified 5-fold cross-validation.

The second dataset was labelled by Ivanov et al. [15], from the PURE dataset developed by Ferrari et al. [16]. They manually extracted 7,745 sentences from 30 of the 79 natural language requirements documents where 4,145 sentences are requirements and 3,600 are non-requirements. To improve labelling accuracy, they employed a manual labelling process by a subject-matter expert and independently verified by additional experts, and any disputed data was removed. We consider this dataset as a measure against imbalanced and small amounts of data.

For the first set of experiments, discussed in Section 5.1, we use the Dronology dataset in the replication package provided by Bashir et al [3] where the dataset was partitioned into five subsets using 5-fold cross-validation. For the second set of experiments, we merged the subsets to form a single dataset and then combined this merged set with the PURE dataset to enlarge the dataset size and address label imbalance. By combining these two datasets, we obtain 8,124 sentences, which contain 4,244 requirements and 3,880 non-requirements with an average length of 134 words. Of these, we randomly selected 30% as the testing set (2,438 sentences), while the remaining 70% was used for fine-tuning (5,686 sentences).

5.3. Results and Discussion

Tables 2 and 3 show our experimental results for all the approaches. The *Weighted Avg.* and *Macro Avg.* columns are the weighted average and macro average of each evaluation metric. The *Training Time* column indicates the total training for each approach. We do not report training times for the two ensemble systems because no training is necessary for these systems. The results for BERT are obtained from our replication of *bert-base-uncased* on the Dronology dataset using stratified five-fold cross-validation and the same configuration used by Bashir et al. [3], to the extent that we could recreate the configuration based on the paper.

RQ1: As shown in Table 2 where we utilized the stratified five-fold cross-validation to evaluate all the models, DeBERTa achieved the best performance with a 82.87% macro F1 score. This

Table 3
Experimental Results for Accuracy and Training Time on the Dronology and PURE datasets

Model	Accuracy	Weighted Avg.			Macro Avg.			Training Time
		Precision	*Recall*	*F1*	*Precision*	*Recall*	*F1*	
DeBERTa	0.9135	0.9135	0.9135	0.9134	0.9135	**0.9128**	**0.9131**	0:20:43
Llama2	0.8970	0.8971	0.8970	0.8971	**0.8969**	0.8971	0.8970	2:24:08
Few-shot Learning	0.7621	0.7628	0.7621	0.7622	0.7622	0.7625	0.7620	0:06:01
DeBERTa + Llama2	**0.9479**	**0.9711**	**0.9479**	**0.9553**	0.8931	0.9121	0.8953	N/A
DeBERTa + Llama2 + Few-shot Learning	0.9286	0.9637	0.9286	0.9398	0.8736	0.8867	0.8707	N/A

raises the prospect that DeBERTa will outperform BERT if one could achieve the same level of performance as Bashir et al. have observed with BERT.

In relation to Llama2 and few-shot learning, we note that these models show lower performance than BERT. This deficit can be attributed to several factors. First, the small size of the training set, with only 304 instances per fold, might be insufficient for these models to effectively learn the task, particularly for Llama2, which typically performs better with larger datasets. Second, the lack of context in the data can be detrimental for requirements classification tasks, as context plays an important role in determining whether a sentence is a requirement or not. Without sufficient context, models may struggle to distinguish between neutral sentences and actual requirements, leading to performance degradation. Third, the training data selection strategy used in the few-shot learning approach could be suboptimal if the selected instances are not representative of the entire dataset or do not cover the diversity of classes and patterns present in the data. In future work, to mitigate these issues, one could consider solutions that increase the size of the training set, incorporate context information, and employ more sophisticated training data selection strategies for few-shot learning, such as clustering-based techniques, manual curation, or active learning approaches.

Regarding training time, compared to BERT, which took 11 minutes to train, DeBERTa and Llama2 required significantly more training time, at 1 hour and 7 minutes and 1 hour and 8 minutes respectively. Therefore, while DeBRETa exhibits better performance, the substantially longer training time suggests that careful consideration needs to be given to the accuracy versus time trade-off.

RQ2: The few-shot learning method exhibited lower performance in both sets of experiments, suggesting that it is unable to match the capabilities of the other models we examined for the requirements demarcation task. However, considering that it requires only 0.8% of the training data (48 used by few-shot learning compared to 5686 used by DeBERTa and Llama2) and has a relatively shorter training time compared to other models, we believe it is a viable option to explore when one has to cope with very small training data.

RQ3: We first attempted to construct an ensemble system by combining the predictions of DeBERTa and Llama2, each weighted at 0.5. As shown in Table 3, ensembling DeBERTa and Llama2 models improves upon individual models in terms of overall accuracy and weighted scores, achieving higher accuracy (94.79%) and weighted scores compared to the individual models. The ensemble system's high weighted F1 of 95.53% indicates its ability to correctly classify the majority class instances. However, it has lower macro scores (precision, recall, and f1) compared to the individual DeBERTa and Llama2 models, indicating the weaknesses of DeBERTa and Llama2 in handling minority classes may be amplified in the ensemble. Finally, we note that adding the few-shot learning model to the ensemble did not provide significant improvements, potentially due to its lower individual performance. The second ensemble system, which includes DeBERTa, Llama2, and few-shot learning, shows a slightly lower accuracy of 92.86% compared to the first ensemble system, with lower weighted and macro scores.

Overall Conclusion. In conclusion, our experiments demonstrated that DeBERTa outperformed other models, albeit requiring a longer training duration compared to BERT. As the volume of training data increased, Llama2 and few-shot learning exhibited comparable performance with DeBERTa. Moreover, the results revealed that while ensemble systems generally enhanced overall performance, they faced challenges in accurately classifying minority classes. Moving forward, it is crucial to carefully evaluate class imbalance and ensemble architectures to address this particular issue.

5.4. Threats to Validity

Internal Validity. Due to resource limitations, our hyperparameter tuning was not exhaustive, meaning there could still be undiscovered model configurations that improve performance. Furthermore, despite the labelled data having been preprocessed and deemed to be of high quality, it is possible that there still remain unusual, anomalous, or improperly labelled examples within the combined requirements dataset. Such data outliers and noise could skew model performance. In future work, robustly detecting and removing possible labelling errors through outlier analysis and visual data profiling could increase data integrity.

External Validity. While our evaluation yielded rather conclusive results on our dataset, the question of whether these findings would generalize to different datasets and varied criteria for requirements demarcation necessitates further case studies.

Construct Validity. Currently, our models only classify text into binary *requirement* and *non-requirement* categories. However, real-world specifications contain a diverse array of semantic types, such as constraints, assumptions, and metadata descriptors. In complex documents, many such more nuanced labels can exist. By training and evaluating solely on a binary classification task, our models may not capture the full richness within specifications. Expanding the label set beyond binary categories could better measure model capabilities and effectiveness for realistic applications.

6. Conclusion

This paper benchmarked two large pre-trained language models, DeBERTa and Llama2, as well as a RoBERTa-based few-shot learning model, for automatically demarcating software and systems requirements in technical specifications. In future work, we plan to create broader training data covering more domains to address the limitations in domain transfer. Furthermore, the integration method used in the ensemble system could be further improved to better take advantage of the complementary traits of DeBERTa and Llama2.

References

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

[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: Open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288 (2023).

[3] S. Bashir, M. Abbas, M. Saadatmand, E. P. Enoiu, M. Bohlin, P. Lindberg, Requirement or not, that is the question: A case from the railway industry, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2023, pp. 105–121.

[4] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, E. Vaz, A machine learning-based approach for demarcating requirements in textual specifications, in: 2019 IEEE 27th International Requirements Engineering Conference (RE), IEEE, 2019, pp. 51–62.

[5] R. C. Staudemeyer, E. R. Morris, Understanding lstm–a tutorial into long short-term memory recurrent neural networks, arXiv preprint arXiv:1909.09586 (2019).

[6] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the association for computational linguistics 5 (2017) 135–146.

[7] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[8] J. Winkler, A. Vogelsang, Automatic classification of requirements based on convolutional neural networks, in: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), IEEE, 2016, pp. 39–45.

[9] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv preprint arXiv:1609.08144 (2016).

[10] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).

[11] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, O. Pereg, Efficient few-shot learning without prompts, arXiv preprint arXiv:2209.11055 (2022).

[12] J. Cleland-Huang, M. Vierhauser, S. Bayley, Dronology: An incubator for cyber-physical system research, arXiv preprint arXiv:1804.02423 (2018).

[13] P. He, X. Liu, J. Gao, W. Chen, Deberta: Decoding-enhanced bert with disentangled attention, arXiv preprint arXiv:2006.03654 (2020).

[14] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685 (2021).

[15] V. Ivanov, A. Sadovykh, A. Naumchev, A. Bagnato, K. Yakovlev, Extracting software requirements from unstructured documents, in: International Conference on Analysis of Images, Social Networks and Texts, Springer, 2021, pp. 17–29.

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