A Transformer Based Pipeline for Software Requirements Classification

Maheen Mashrur Hoque
Islamic University of Technology
Gazipur, Bangladesh
maheenmashrur@iut-dhaka.edu

Arman Hossain Dipu
Islamic University of Technology
Gazipur, Bangladesh
armanhossain@iut-dhaka.edu

Ahsan Habib
Islamic University of Technology
Gazipur, Bangladesh
ahsanhabib@iut-dhaka.edu

Ajwad Abrar
Islamic University of Technology
Gazipur, Bangladesh
ajwadabrar@iut-dhaka.edu

Abstract

The automation of the software development lifecycle (SDLC) is a key challenge in the field of software development. One of the main challenges in automation of the SDLC lies within the work of collecting, analyzing and establishing requirements, where the requirements collected from the stakeholders are often noisy, which can be difficult to organize. In order to facilitate the automation of requirement analysis, our study proposes a cost-effective approach - leveraging the lightweight DistilBERT and RoBERTA models with a method called "Ensemble Pooling" to filter out relevant requirements. Our experiments showcased an accuracy of 78% for the "Ensemble Pooling" method, a higher score than readily available sequence classifier versions of the aforementioned models.

CCS Concepts

• **Applied computing → Document analysis**.

Keywords

Software Requirements, Software Development Lifecycle (SDLC), Classification, Natural Language Processing (NLP), Transformers,Ensemble

ACM Reference Format:
Maheen Mashrur Hoque, Arman Hossain Dipu, Ahsan Habib, and Ajwad Abrar. 2018. A Transformer Based Pipeline for Software Requirements Classification. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Software requirements can be thought of very high level description of what a software system is expected to do in order to solve a business requirements and are the reflection of the expectations of the stakeholders and clients [13]. A requirement should have clarity, be consistent and have completeness [20]. However, factors like communication gap, difference in perspective, and complexity of scope can make requirements vague and difficult to comprehend for the developer [12]. Due to these factors, requirements from the stakeholders may often contain irrelevant information (which we are referring to as "noise"), which can later on cause difficulties to analyze those requirements (for example, discerning functional and non-functional requirements). This causes bottleneck in the SDLC as system architects spend much time organizing the requirements. The challenge of requirements organization can be classified as a natural language processing (NLP) problem, with the process of filtering actual requirements from the irrelevant texts being the first step. Low infrastructural cost is also a factor of consideration here. So, our research objective in this work can be defined as - "Noise reduction via classifying relevant and irrelevant requirements, through cost-effective means", with the works of Ivanov et. al. [7] acting as baseline. The experiments, dataset and results are discussed in later sections.

2 Literature Review

This section provides a comprehensive descriptions of the previous works that were reviewed for this research. In order to avoid parochial view over the research objective, the reviews were not limited to only language model based approaches.

2.1 Esoteric Approach

The works of Siahaan and Darnoto distinguishes irrelevant requirements using the MultiPhiLDA method [16]. Their method works by distinguishing topic–word distribution of actor words and action words, governed by polynomial probability functions, essentially making it a statistical distribution analysis.

Another method by Alharbi et. al. leverages semantic embeddings to filter ambiguous requirements [1]. The embedding were fine-tuned to the specific task, allowing it to capture context-specific nuances related to ambiguity.

Similarly, the works pf Malik et. al. leverages cosine similarity over sentence embeddings to classify conflicting requirements [10]. Although, not directly related, this work provided us insight on how to levarage similarity scores.

Another approach by Norabid et. al., is to define Entity-Relation Extraction Rules for performing linguistic analysis to extract dependency relations and part-of-speech (POS) in formation from a given text [11]. These rules guide the extraction of entities and

Conference acronym 'XX, Woodstock, NY
© 2018 ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

relations from the text. Final step is Triple Extraction and MKG Construction. A triple extractor is developed, incorporating the rules. The extractor extracts triples (subject-relation-object) from the news articles dataset. These triples form the basis of this MKG based study of the authors.

2.2 Language Model Approach

A language model based approach by Ivanov et. al. for extracting requirements from unstructured text uses BERT [4] model fine-tuned with a manually annotated dataset from the PURE (Public Requirements) corpus [6]. They achieved an accuracy of 74%, compared to two other baseline approaches – fastText (open source natural language processing toolkit) with an accuracy of 60%, and ELMo (a foundational text-embedding model) paired with SVM (Support Vector Machine classifier) with an accuracy of 59%. Due to the similarity of objective, this work was considered as a baseline for us to compare against.

3 Dataset

For our work the, the PURE (Public Requirements) dataset [6], created by Ferrari et. al., was the primary source of data for the experiments of RO1. This contains 79 software requirements document (SRS) that contains 34,286 sentences. However, the dataset does not provide any additional labelling for the requirements sentences to aid natural language processing tasks. Some of the previous works, notably [7] and [16] utilized this dataset in their works via labeling in various ways.

A labeled subset of the PURE dataset, called the RegExpPURE Dataset by Ivanov et. al. [7] was used in our studies. This dataset has proper balancing in it's data, with 2474 not requirements and 2832 requirements in train, 467 not requirements and 1058 requirements in test and 255 not requirements and 650 requirements in validation set, making it a suitable training candidate. Note that these same train, test, and validation sets were used to get a more comparable experiment setup to compare with the baseline work.

4 Methodology

From the discussions of possible approaches for NLP tasks concerned with primarily extraction and classification presented in section 2, it was decided to approach by leveraging transformers models. Note that much of our approach is uni-modal, only working with text based data. As of our current advancement and the nature of how requirements descriptions usually delivered to the development personnel and stakeholders, it is a reasonable assumption that utilizing text based data would be sufficient. The future extension of our work may be benefited from using a multimodal approach, similar to the works of Norabid et. al. [11].

4.1 Choice of Language Models

Here, we utilized two lightweight language models, based on the Transformers architecture - DistilBERT and RoBERTa.

For natural languages, it is quite common that the correlation amongst the words in the context of the expression of that sentence can showcase long range dependencies. The self-attention mechanism allows Transformer models to capture long-range dependencies more effectively than RNNs and LSTMs, which struggle with long-term context due to their sequential nature [19]. Moreover the architecture of Transformer models is highly versatile and can be adapted to a wide range of NLP tasks, including the classification task in this work.

DistilBERT is a lightweight variant of the BERT model which retains almost 90% of the BERT's performance in various NLP benchmarks [14]. DistilBERT uses model compression (student-teacher approach) to reduce layers from BERT to half while matching the logits, hidden states and attention distribution. The reason why we chose this model for our work is the efficiency that DistilBERT showcases over BERT. DistilBERT requires lower memory, which makes it ideal for anyone to utilize this approach of this study without any significant computing requirements. On the other hand, RoBERTa is an optimized version of BERT that is trained on a larger text corpus than that of BERT and also uses dynamic masking during pre-training, which changes the masking pattern of the input sequence at each epoch, allowing the model to see a larger variety of contexts [9]. Moreover, RoBERTa retains the same architecture as BERT but optimizes hyperparameters, including removing the Next Sentence Prediction (NSP) objective. This modification improves the efficiency of the training process and the model's overall performance [9]. Due to these reasons, RoBRETa outperforms BERT outperforms BERT in many benchmarks and is a strong yet lightweight candidate for classification tasks. Hence, it was also chosen in our experiment.

4.2 Definition of Noise in Requirements

In the context of software requirements, noise can be referred to as irrelevant information or non-essential details that do not contribute to the actual functionality or goals of the system. These noises can be redundant information, ambiguous language or non-requirements. To illustrate an example from the RegExpPURE dataset [7], the following is a clear requirement:

> System Initialization shall [SRS014] initiate the watchdog timer.

And the following is a noise:

> A Working Group was established in May, 2006 to develop high-level functional specifications for an NLM Digital Repository for NLM collection materials and to identify policy and management issues related to the creation, design and management of the repository.

We can see that in order to recognize what is a requirement and what is not a requirement, one needs to read the entire sentence and map the terms of the sentences to clarify what the sentence expresses. This can be used as an analogy of what the term 'long range dependency' entails for transformer models, the model's awareness of the expressive meaning of every term in the sequence with one another.

4.3 Ensemble Pooling Approach

For this classification task of filtering noise, we propose a method that we call "Ensemble Pooling". Our approach consists of three layers -

(1) **Tokenizer Layer :** This layer will appropriately tokenize the input sequence from text to corresponding vector representation.
(2) **Transformer Layer :** This layer will produce the contextual embedding for the input sequence by feeding into the vector representation of the tokenizer layer.
(3) **Classifier Layer :** This layer will use the contextual embedding (via pooling) produced by the transformer layer to classify the requirements from the noise. This layer would be trained with appropriate labels for classification. In our experimentation, we got better result with ensemble classifiers (hence the name, 'Ensemble Pooling').

The following Figure 1 represents the architecture of our approach.

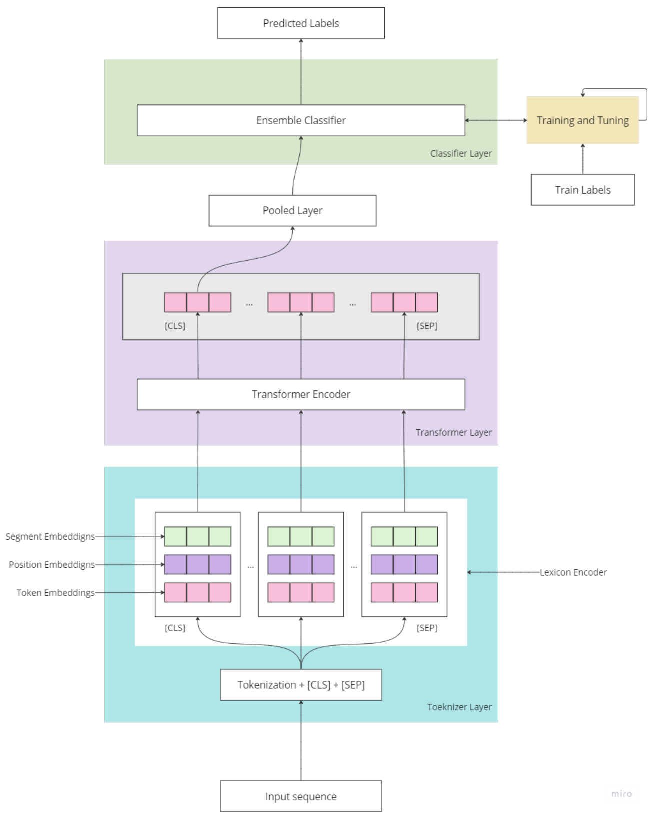

Figure 1: The Ensemble Pooling Classification Approach

Here, we utilized the last hidden layer of the transformer models, which contains integrated information or the final contextualized embeddings from the entire sequence in a multidimensional vector representation (in the Huggingface Transformer library [14], the dimension is $(Seq_{size} \times Seq_{length} \times Size_{hidden})$ the default hidden size is 768). But the most important content of the layer is the [CLS] token representation, which is added to each of the embeddings vector by the tokenizer and the vector for that token acts as the aggregate representation of the entire sequence [4] [15], that we use with the ensemble classifiers for classification tasks. The DistilBertForSequenceClassification and RobertaForSequenceClassification (based on DistilBERT and RoBERTa) classifiers offered by the Transformers library uses a similar approach, but with linear classification layer on top. These were compared as baselines.

Note that the transformer layer does not have any training cycle. This is because the transformer models used here (BERT and RoBERTa) are already trained over a large corpus of the English language and already have the necessary vocabulary structuring for language related tasks [14] [9]. This facilitates in reducing the cost and time that one had to invest with training.

4.4 Classification Layer

The classification layers were trained over the pooled vectors from the transformer layer and the labels from the dataset. The labels were converted to 0 for not a requirement (i.e. noise) and 1 for requirement (i.e a proper requirement). Then 4 classifier models were trained and evaluated, which were Random Forest Classifier, XGB Classifier, LightGBM classifier and SVM Classifier. Out of these, 3 are ensemble models (Random Forest, XGB, LightGBM). We chose ensemble models because they leverage multiple learning algorithms (or sub-classifiers) to obtain better predictive performance than could be obtained from any of the standalone models. Moreover, due to combining sub-classifier, ensemble models are robust against overfitting [2]. On the other hand, SVM is not an ensemble model, rather it is a standalone classifier that determines a hyperplane to best separate the datapoints into classes [3]. SVM was also experimented with as it had shown good performance in NLP related classification tasks in several works [5] [18] [8].

4.5 Training Parameters

For the experiments, the core DistilBERT and RoBERTa models, used for ensemble pooling were not trained specifically for this study. However the two transformer based classifiers and the other classifiers were trained. The transformer based classifier models were fine tuned for classification, with the hyperparameters being the same for both - Batch Size = 16, Max Length = 128, Learning Rate = 2×10^{-5} and was trained over 10 epochs. Adam optimizer was used and Cross Entropy Loss was the loss function for both of the models. Both DistilBERT and RoBERTa is based on the BERT model. Their parameters have no difference. Where they do differ is in how they handle sequences in their inner layers (which is discussed in section 4)

For the classifiers, all were fine tuned with 10 fold grid search cross validation. The target metric for the hyperparameter tuning was accuracy. The optimal hyperparameters for the models are as follows in Table 2, Table 1, Table 4, and Table 3.

Again, the term "Pairing" is referred to as using the hidden layer of the language models through the classifier models to make predictions.

5 Results

As per the setup described in section 4, we compared our experiments with the baseline paper. Similar to the baseline work, results are based on the validation set. For the result comparison, primarily used the accuracy metric (The rationale behind this choice of metric

Table 1: Random Forest Hyperparameters for RoBERTa and DistilBERT Pairings

Hyperparameter	**Model Pairing**	
	RoBERTa	***DistilBERT***
Max Depth (`max_depth`)	None	10
Min Samples Leaf (`min_samples_leaf`)	4	4
Min Samples Split (`min_samples_split`)	2	2
Number of Estimators (`n_estimators`)	300	100

Table 2: LightGBM Hyperparameters for RoBERTa and DistilBERT Pairings

Hyperparameter	**Model Pairing**	
	RoBERTa	***DistilBERT***
Colsample by Tree (`colsample_bytree`)	1.0	1.0
Learning Rate (`learning_rate`)	0.1	0.1
Min Child Samples (`min_child_samples`)	20	50
Number of Estimators (`n_estimators`)	300	200
Num Leaves (`num_leaves`)	50	50
Subsample (`subsample`)	0.8	0.8

Table 3: XGBoost Hyperparameters for RoBERTa and DistilBERT Pairings

Hyperparameter	**Model Pairing**	
	RoBERTa	***DistilBERT***
Learning Rate (`learning_rate`)	0.1	0.1
Max Depth (`max_depth`)	5	5
Number of Estimators (`n_estimators`)	300	300
Subsample (`subsample`)	0.8	0.7
Colsample by Tree (`colsample_bytree`)	0.8	1
Gamma (`gamma`)	0.1	0

Table 4: SVM Hyperparameters for RoBERTa and DistilBERT Pairings

Hyperparameter	**Model Pairing**	
	RoBERTa	***DistilBERT***
C (`C`)	10	10
Gamma (`gamma`)	scale	auto
Kernel (`kernel`)	rbf	rbf

is discussed in the section 6). The Table 5 contains the results and the Table 6 shows the baseline paper result.

For our experiments and proposed approach of identifying requirement noise, we used the RegExpPURE dataset [7]. So, our baseline was the results in that paper for the dataset in which they utilized the BERT base model for extracting requirements. Though their work may not directly state noise reduction, but their work achieved similar objectives. As seen in section 3 their dataset consists labeled sentences of requirements and non requirements. And these non requirement sentences are precisely what we have defined as noise. So, the rationale for using this paper as a baseline is quite reasonable.

Table 5: Results Comparison of the Various Approaches

Approach Category	**Model / Technique**	**Accuracy (%)**
DistilBERT	DistilBERT Sequence Classifier	76
	DistilBERT + Random Forest	**78**
	DistilBERT + XGB	77
	DistilBERT + LightGBM	77
	DistilBERT + SVM	76
RoBERTa	RoBERTa Sequence Classifier	77
	RoBERTa + Random Forest	75
	RoBERTa + XGB	77
	RoBERTa + LightGBM	77
	RoBERTa + SVM	77

Table 6: Results of the Baseline Paper [7]

Approach	**Accuracy (%)**
fastText	60
ELMo + SVM	59
BERT	74

6 Discussion

6.1 The Rationale for Accuracy

Accuracy is intuitive and easy to understand. It directly reflects the overall correctness of the model's predictions. In many scenarios, balanced performance across different classes is desirable. Accuracy treats all classes equally. Moreover, when the class distribution is roughly uniform (which is the case for the RegExpPURE Dataset), accuracy is a reasonable choice. And hence, accuracy was the chosen metric. [17].

6.2 Base Classifier vs Ensemble Pooling

From the results in Table 5 and Table 6 shows that our approach combining DistilBERT and Random Forest in ensemble pooling performs better in terms of accuracy when compared against all the approaches and baseline results. However, all the approaches with RoBERTa had quite close results, except Random Forest. This deviation is most likely due to how RoBERTa utilizes byte-pair encoding in tokenization and how the inner encoder layers attune to the context of the sequences [9].

The baseline paper showed the most promising result with BERT with about 74% in accuracy. But baseline classifier of BERT, similar to DistilBERT and RoBERTa uses a neural network layer with a linear softmax activation function to generate class probability distribution. While effective in most cases, this classification task can benefit from more robust method of classification, ensemble classification. Since ensemble classifiers use bagging of sub classifiers, it is more robust in classification. Random Forest is one such ensemble method that ensembles decision trees. These trees or sub classifiers work in tandem for classification and improves accuracy, as seen from our experiments. Again, the rationale behind choosing DistilBERT and RoBERTa comes to their lightweight performance. They require much less computing resources compared to some other larger models and still remains a popular choice for smaller NLP related works.

However, the lightweight nature is also one of the drawbacks of our approach. The accuracy results are still in the range below 80, which is a clear indication that these models fall short due to having smaller number of parameters compared to state of the art GPT, Gemini and LLAMA models (inferred from the "Densing law of LLMs" [21]). So, as a future extension in requirement noise reduction, the viability of the state of the art models as well as stronger classification layers (stacking multiple classifiers for starters) remain as an yet to explore prospect.

7 Conclusion

In conclusion, this study contributed valuable insights into the application of transformer models in natural language processing tasks, specifically in the context of software requirements classification. By focusing on noise reduction, this research lays the foundation for enhancing the efficiency and effectiveness of the software development process. Our developed methodology showcased improved performance when compared with various out-of-the-box classifiers in terms of accuracy.

While the current work primarily addressed noise reduction in requirements, the future direction of this research involves extending the classification pipeline to encompass all research objectives, including improving requirement assessment, conflict resolution and multimodal analysis (involving charts, diagrams, wireframes - etc.).

References

[1] Sadeen Alharbi. 2022. Ambiguity Detection in Requirements Classification Task using Fine-Tuned Transformation Technique. In *CS & IT Conference Proceedings*, Vol. 12. CS & IT Conference Proceedings.

[2] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[3] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20 (1995), 273–297.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Harris Drucker, Donghui Wu, and Vladimir N Vapnik. 1999. Support vector machines for spam categorization. *IEEE Transactions on Neural networks* 10, 5 (1999), 1048–1054.

[6] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. Pure: A dataset of public requirements documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 502–505.

[7] Vladimir Ivanov, Andrey Sadovykh, Alexandr Naumchev, Alessandra Bagnato, and Kirill Yakovlev. 2021. Extracting software requirements from unstructured documents. In *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 17–29.

[8] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, 137–142.

[9] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[10] Garima Malik, Mucahit Cevik, Devang Parikh, and Ayse Basar. 2022. Identifying the requirement conflicts in SRS documents using transformer-based sentence embeddings. *arXiv preprint arXiv:2206.13690* (2022).

[11] Idza Aisara Norabid and Fariza Fauzi. 2022. Rule-based text extraction for multimodal Knowledge Graph. *International Journal of Advanced Computer Science and Applications* 13, 5 (2022).

[12] Bashar Nuseibeh and Steve Easterbrook. 2000. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. 35–46.

[13] Klaus Pohl. 2016. *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant*. Rocky Nook, Inc.

[14] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[15] Justyna Sarzynska-Wawer, Aleksander Wawer, Aleksandra Pawlak, Julia Szymanowska, Izabela Stefaniak, Michal Jarkiewicz, and Lukasz Okruszek. 2021. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research* 304 (2021), 114135.

[16] Daniel Siahaan and Brian Rizqi Paradisiaca Darnoto. 2022. A Novel Framework to Detect Irrelevant Software Requirements Based on MultiPhiLDA as the Topic Model. In *Informatics*, Vol. 9. MDPI, 87.

[17] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information processing & management* 45, 4 (2009), 427–437.

[18] Kentaro Torisawa et al. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 315–324.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[20] Karl E Wiegers and Joy Beatty. 2013. *Software requirements*. Pearson Education.

[21] Chaojun Xiao, Jie Cai, Weilin Zhao, Biyuan Lin, Guoyang Zeng, Jie Zhou, Zhi Zheng, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. Densing law of llms. *Nature Machine Intelligence* (2025), 1–11.