A Transformer Based Pipeline for Software Requirements Classification

1st Maheen Mashrur Hoque
Department of Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
email address or ORCID

2nd Arman Hossain Dipu Khan
Department of Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
email address or ORCID

3rd Ahsan Habib
Department of Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
email address or ORCID

4th Ajwad Abrar
Department of Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
email address or ORCID

***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 two cost effective approaches, the first being leveraging the lightweight DistilBERT and RoBERTA models with a method called "Ensemble Pooling" to filter out relevant requirements, and the second one being a retrieval augmented generation (RAG) based classification leveraging GPT-4 and ChromaDB vector store to discriminate between functional and non-functional requirements. The experiments conducted by us showcased an accuracy of 78% for the first approach and 86.67% in the second approach.**

***Index Terms*—Software Requirements, Software Development Lifecycle (SDLC), Classification, Natural Language Processing (NLP), Transformers, Retrieval Augmented Generation (RAG), Ensemble**

I. 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 [1]. Software requirements are typically divided into three categories [2] – Functional (Specify the purpose of the system), Non-Functional (Define the system's quality attributes such as performance), and Domain Requirements (Specific to the context of the application domain such as memory requirement). A requirement should have clarity, be consistent and have completeness [3]. However, factors like communication gap, difference in perspective, and complexity of scope can make requirements vague and difficult to comprehend for the developer [4]. Due to these factors, requirements from the stakeholders may often contain irrelevant information (which we are referring to as "noise"), and it can be difficult to discern 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. Hence, our research objective (RO) in this work is to deal with two critical steps:

- **RO1:** Noise reduction via classifying relevant and irrelevant requirements.
- **RO2:** Improving requirement assessment via functional and non-Functional requirements.

In both objectives, we experimented with two different approach for classification, with the works of Ivanov et. al. [5] acting as baseline. The experiments, dataset and results are discussed in later sections.

II. 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.

A. Esoteric Approach

The works of Siahaan and Darnoto distinguishes irrelevant requirements using the MultiPhiLDA method [6]. 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 [7]. 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 [8]. Although, not directly related, this work provided us insight on how to levarage similarity scores.

B. Language Model Approach

A language model based approach by Ivanov et. al. for extracting requirements from unstructured text uses BERT [9] model fine-tuned with a manually annotated dataset from the PURE (Public Requirements) corpus [10]. 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.

III. Dataset

A. PURE Dataset

For our work the, the PURE (Public Requirements) dataset [10], 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 [5] and [6] utilized this dataset in their works via labeling in various ways.

A labeled subset of the PURE dataset, called the Reg-ExpPURE Dataset by Ivanov et. al. [5] 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.

B. FR-NFR Dataset

For the experiments of RO2, a custom dataset was made with SRS documents from 3 real projects conducted by the Ministry of ICT, The People's Republic of Bangladesh - A Training Database, an Enterprise Management System, and an AI Chatbot. The requirements were collected

IV. Ease of Use

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

V. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections **??–??** below for more information on proofreading, spelling and grammar.

A. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

B. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

C. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

References

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

[2] I. Sommerville, *Software Engineering, 9/E*. Pearson Education India, 2011.

[3] K. E. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.

Fig. 1. Example of a figure caption.

[4] B. Nuseibeh and S. Easterbrook, "Requirements engineering: a roadmap," in *Proceedings of the Conference on the Future of Software Engineering*, 2000, pp. 35–46.

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

[6] D. Siahaan and B. R. P. Darnoto, "A novel framework to detect irrelevant software requirements based on multiphilda as the topic model," in *Informatics*, vol. 9, no. 4. MDPI, 2022, p. 87.

[7] S. Alharbi, "Ambiguity detection in requirements classification task using fine-tuned transformation technique," in *CS & IT Conference Proceedings*, vol. 12, no. 21. CS & IT Conference Proceedings, 2022.

[8] G. Malik, M. Cevik, D. Parikh, and A. Basar, "Identifying the requirement conflicts in srs documents using transformer-based sentence embeddings," *arXiv preprint arXiv:2206.13690*, 2022.

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

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