ARQM: A Tool for Requirements Quality Analysis

Dylan Porter, Joanna F. DeFranco, Joseph Reid, Everton Guimaraes, Satish Srinivasan, Phillip A. Laplante

Corresponding e-mails: burghporter314@gmail.com; jfd104@psu.edu; jrr5374@psu.edu ; ezt157@psu.edu; sus64@psu.edu; pal11@psu.edu;

Abstract

Requirement engineering (RE) consists of complicated processes and is often prone to human error. In response, tools were built to address RE tasks such as requirement identification, screening, management, and analysis. Although many tools have been successful at analyzing requirements, existing tools have relied upon textual heuristics or outdated methods to perform automated requirement tasks. Modern Artificial Intelligence (AI) and Machine Learning (ML) methods permit a more holistic ability to perform requirements screening tasks. Utilizing modern methods, we propose the Automated Requirements Quality Management (ARQM) tool to help with requirement analysis (i.e., requirement identification/qualification and requirement quality). ARQM allows for both requirement screening and quality analysis based on the IEEE 29148:2018 standard. This study used the results of models among human annotated datasets for both requirement screening and requirement quality analysis. The results indicate that modern AI methods can learn general patterns for both requirement screening and quality analysis with certain limitations. Further, the ARQM tool utilizes Explainable Artificial Intelligence (XAI) to help visualize the violations detected by the various AI models.

Keywords: Requirements Screening, Requirements Quality Analysis, AI/ML, IEEE 29148

1. Introduction

Due to heightened complexity of systems and their associated quality requirements, the requirement engineering (RE) process becomes increasingly more difficult to manually implement. Even with the added system complexity, RE activities help to facilitate an understanding between the users and developers of a system, making RE crucial for project success and adoption.

More importantly, failure to adequately engage in all forms of the RE process may lead to costly delays and substantial system issues among product features. Ultimately, RE processes can help mitigate associated delays, while improving product quality in relation to user features. Although engaging in most requirement processes can prove substantially beneficial, these activities can often be challenging, time-intensive, and prone to human error [1].

To achieve the holistic functions of RE, tasks typically involve analysis, management, elicitation, modeling, and verification of the system requirements. Ultimately, each of these RE tasks requires extensive resources to properly complete manually. To mitigate these efforts, automated tools have sought to provide users with task efficiency. These automation efforts help support the broad requirement engineering tasks by minimizing human effort by implementing algorithms for key analytical components of each task. Naturally, the implementation methods and focus of these tools differ drastically.

Based on existing tool categories, the Automated Requirements Quality Management (ARQM) tool primarily focuses on requirement screening (i.e., if a statement is in fact a requirement) and requirement quality analysis (i.e., does it follow IEEE 29148 standard for definitions and quality reference). Requirement screening serves as a necessary preprocessing step for most of the automated tasks; ARQM implements requirement screening as a preliminary step for other analytical tasks such as requirement quality analysis. For each requirement task automation focus, differing methods such as rule-based and AI-based methods are utilized among various existing tools. The ARQM application extends upon these existing implementations through the utilization of modern Natural Language Processing (NLP) and AI-based methods [2], [3].

Even with the implementation of individual requirement automation tools, there exists a serious practical gap for simplistic requirement document processing. The functionality gap of practical tools manifests in many forms, including lack of modern algorithmic utilization, disparate tool reliability, lack of tool support, and theoretical implementation saturation. Primarily, existing implementations do not utilize practitioner feedback. The ARQM tool extends upon existing implementations provided by past tool implementations, with an emphasis on the NASA Automated Requirements Measurement (ARM) and reconstructed tools, to provide an automated requirement analysis (Carlson & Laplante, 2014). The ARQM tool works without human input and provides a human-readable analysis document after document processing, simplifying the process of requirement quality analysis [1], [3], [4], [5].

As part of the screening and quality analysis process, the ARQM tool focuses primarily on the IEEE 29148 standard for definitions and quality reference. The standard provides templates and definitions of what constitutes requirements and potential quality violations. The standard also specifies quality attributes for both requirements and requirement groups that outline the ideal traits constituting quality requirement text. Although many studies provide different quality models, the IEEE 29148 standard helps to consolidate the analysis of requirements to a specific process that relates to RE and the associated activities. For the analysis of quality attributes, the research focuses on understanding different levels of linguistic focus, including lexical, syntactical, semantical, and pragmatic analysis [6].

To accomplish the requirement screening and analytical tasks, the ARQM tool focuses on different model implementations and methods to maximize results among both publicly available and human-annotated datasets. For these datasets, the requirement screening process primarily utilized the PUblic REquirements (PURE) dataset to achieve accurate model inference while the requirement quality analysis task utilized a human-annotated dataset, derived from multiple requirement documents, to tune various models for ideal inference [7]. The PURE dataset serves as a benchmark for requirement screening; the dataset is comprised of different sources and representations of requirements.

Additionally, the PURE dataset contains thousands of entries for requirement and non-requirement representations, allowing for the training of AI models. Lastly, the quality analysis process focused on a select set of individual quality attribute criteria based on their analytical complexity. The use of Explainable Artificial Intelligence (XAI) helped to uncover the reasoning behind model inference while providing inline feedback to the user for the generation of quality result artifacts. The remainder of this paper is organized as follows: Section 2 introduces definitions and key concepts; Section 3 reviews related work; Section 4 presents the ARQM tool; Section 5 describes the validation methodology and results; Section 6 discusses implications and limitations; and Section 7 concludes.

2. Background and Key Concepts

2.1. Requirement Screening

ARQM automates both requirement screening and requirement quality analysis. As part of the automation task, requirement screening consists of identifying all requirement-related text within a document; depending on the document, requirements are generally represented through different syntaxes, making automatic screening difficult. As an added consideration of requirement screening , requirement length, requirement type, and relevance all matter during the requirement automation process. The IEEE 29148 standard provides typical standards associated with requirements, while other works show the various representations of requirements through different document formats [6], [8].

2.2. IEEE 29148 Standard Requirement Quality

According to the IEEE 29148 standard, analyzing requirement quality consists of adherence of certain quality attribute factors across individual requirements and requirement groups. This specification outlines critical quality criteria that all requirement text should follow, serving as a benchmark for requirement quality. Quality attributes such as Ambiguity, Completeness, and Consistency look at certain aspects of requirement quality; defects in these quality attributes can lead to misunderstandings or development issues and even project failure for a system. Although a single violation does not necessarily constitute a problematic requirement, quality attribute violations often serve as heuristics for potential issues. These quality attribute criteria also apply to different types of grammatical analysis, making their associated automation more difficult than requirement identification [2], [6].

2.3. Heuristics and Rule-Based Methods

For both requirement identification and quality analysis, common text and phrases can often help with inference. Heuristics and rule-based methods look at commonly used text to help make determinations related to requirement automation. For instance, requirement screening can look for keywords or phrases to help identify potential requirements or non-requirement-related text. For quality analysis, certain words or phrases such as *better than*, *this*, or *it* can serve as heuristics for potential quality issues [6]. Many existing studies focus on concepts like *requirement smells*, a way to utilize heuristics to determine potential quality violations [1], [9]. Primarily, the utilization of heuristics and rule-based methods limits the need of complicated training data, is relatively simple to implement, and tends to be accurate for most levels of analysis [10]. Often, the rule-based implementations of quality analysis are combined with human intuition and only serve as an informative feature of the automated process.

2.4. AI-based Methods

Artificial Intelligence (AI) and Natural Language Processing (NLP) implementations help to perform requirement identification and quality analysis using either more complicated heuristics or complex patterns learned through labeled data [3]; These methods can help better understand and encapsulate the complicated relationships associated with semantics and pragmatic language, making the inferences useful for practical classification tasks. For NLP-based implementations, broader heuristic formulations are possible through methods such as Semantic Role Labeling (SRL), Named Entity Recognition (NER), Cosine Similarity and Part-of-Speech (POS) tagging [11]. Through these NLP-based techniques, broader heuristics are generated that can help understand complicated patterns of language beyond simple lists of words and phrases [3].

For AI-based implementations, unsupervised, semi-supervised, and supervised methods are available to help learn complicated patterns for requirement screening and quality analysis. Unsupervised learning and semi-supervised learning methods can help uncover key features, while also helping to create labeled datasets. Supervised learning methods through deep learning (DL) can help learn complicated, generic patterns through the use of labeled datasets. More advanced methods can help synthesize model knowledge to specific classification tasks, limiting the need for extensive labeled datasets and improving model accuracy [12]. Explainable artificial intelligence (XAI) methods can help uncover the reasons for model inference to help uncover features and explanations [12], [13].

2.5. Requirement Screening Analysis

For requirement screening, the primary question becomes how to meaningfully divide textual units while also optimizing algorithms to differentiate between non-requirement and requirement text. Because there are multiple representations of requirements through different requirement text, model optimization is difficult [6]. Therefore, automation for the ARQM had to include ways to meaningfully divide the document data into potential requirement units, while also screening requirement-related text. To successfully implement these functions, many methods existed to help extract out important features that were representative of requirement text [3].

For the practical implementation of a requirement screening tool, many models and tools exist to help screen requirements [5]. For instance, statistical and light-weight models were used, while more complicated DL models were also implemented to classify requirements. These models learned different representations of requirement text, helping to uncover different features representative of differentiating classes. The requirement screening task primarily focused on finding the ideal models that best identified requirement and non-requirement-related text. Ultimately, the focus of requirement screening was on both the PURE dataset and the human-annotated dataset, allowing for an understanding of how the learned features from the two datasets differentiate based on the model results [14].

2.6. Requirement Quality Analysis

For requirement analysis, challenges with labeled datasets were a primary concern during the practical development of the ARQM tool. To mitigate the problem of minimal data, the ARQM models were trained on a human-annotated dataset across a small set of quality attributes, including *Ambiguity, Feasibility, Singularity,* and *Verifiability* [6]. The simplistic nature of the quality attributes made them prime candidates to test across modern AI-based methods. As an added consideration of the research, the ARQM tool focuses on the ability of AI to practically analyze these quality attributes beyond simple heuristics, with a focus on practical reading of human users. Additionally, the tool looks into potential implementations of XAI to help visualize the quality attribute violation results for intuitive feedback [13].

The overall goal of the ARQM application is to provide practical feedback on quality violations to the user based on useful violation detection. The tool focuses on easy-of-use and clarity for feedback, while also trying to limit manual human annotation and interaction. The multiple iterations of the ARQM tool highlight the simplistic development and user-based focus that allows for minimal human interaction. By eliminating human involvement with the application, the ARQM tool had to overcome key challenges, such as requirement screening, to operate effectively. The ARQM tool also considers the limitations of previous tools and provides an improved implementation for both requirement screening and quality analysis [1], [6].

3. Related works

Understanding the methods, implementations, and practical tools of existing work was pivotal for the development of the ARQM application. Although the focus of the research was primarily on the development of a practical tool, the theoretical work helped to inform the best directions to implement requirement identification and quality analysis. When analyzing existing automated tools, the gap became apparent due to the lack of tool availability and utilization of outdated, rule-based methods. Based on the synthesize of research and an understanding of past tool implementations, the ARQM was developed as a modern, AI-based tool [1].

3.1. Requirement identification

Existing implementations and tools have focused on requirement screening, specifically. As an example, many requirement screening methods focus on requirement classification across various different mediums. A systematic literature review looked at tool implementations that analyzed online text across Stack Overflow to help classify requirements. The study focused on how the various implementations identified functional vs. non-functional requirements, showing the results of different models, with ML-based implementations outperforming for both classifications [15]. Another study focused on construction documents to perform requirement screening, identification, and traceability their tool was able to identify requirements and associate the requirements with external supporting documentation. Their results showed that ML-based models, such as the convolutional neural network (CNN), along with textual embeddings outperformed for identification [16].

As part of the requirement screening process, many studies also look at uncovering metadata for the associated requirement; for instance, understanding requirement relationships and hierarchy is a pivotal step for identifying and classifying requirement quality. Because requirement screening often serves a preliminary processing step to other requirement automation tasks, many implementations look beyond simple identification. For instance, a study implemented a question-answering (QA) approach to help uncover requirement hierarchy within a requirement document. This method recursively identified dependent requirements through the combination of QA and fine-tuning techniques on complex LLMs, allowing for the transfer of model knowledge to an automated requirement engineering task [17]. Other studies implemented extraction of requirement actors, actions, and objects to create a simplified ontology or knowledge base for easier processing. This data simplification technique allowed for the synthesis of complicated requirement information into a formal representation [18], [19].

For rule-based implementations of requirement screening and identification, many studies look at the possibility of textual statistics and language representations to generate important features. An existing implementation looks at textual heuristics that help identify requirements. Primarily, the study looks at token-based, syntactic, semantic, and frequency-based features to help train their associated models. They found that frequent model verbs and verb phrases made the biggest difference during the requirement identification process. Other features such as number of tokens and alphabetics and stative verb presence were also important. The Random Forest model outperformed during model training [20].

Another study utilized unsupervised methods like semantic similarity distance to help identify and extract requirement text from requirement documents. They extend the semantic similarity distance algorithm to non-functional requirement (NFR) identification, helping to identify NFRs across classifications like Availability, Legal, and Scalability. Through threshold similarity, they were able to achieve high results for precision, recall, and F1 across the various classes, even with the difficulty of multi-label classification amongst NFRs [21]. Other methods utilized embedding approaches in conjunction with supervised learning methods. The article discusses a manually labeled dataset for requirement classification along with a confidence score provided by practitioners. Similarly to the previous implementation, the study analyzed NFR classification. With the help of TF-IDF, Word2Vec, and BERT embedding algorithms, the study achieved high precision, recall, and F1 scores of 0.92, 0.91, and 9.2, respectively. Additionally, they utilized a fusion, ensemble-based model, that combined BERT embeddings and a CNN implementation to achieve the best results for classification [22].

For supervised learning, many existing studies have looked at how requirement screening and identification is accomplished through labeled data. For instance, an existing study compared rule-based implementations with ML models. Primarily, the author discusses the performance of various multi-gram word and phrase representations when compared to the results of various ML models. Surprisingly, the author found that rule-based algorithms performed similarly to ML-based algorithms. Although their dataset was limited, their findings indicated that requirement identification could achieve high accuracy through the use of simple heuristics [23]. Another study analyzed more complicated representations of requirements through the use of CNNs and RNNs. The study tried to identify inspection requirements from construction documents as part of the automated requirement process. In the study, they treat the delimiter of requirements as sentences and achieve an accuracy of 91% and 88% for the CNN and RNN models, respectively. During training, the *GloVe* embeddings seemingly helped with the accuracy of model inference [16]

Lastly, other implementations utilized more complicated models and LLMs to achieve -screening of requirements. For instance, a study looked at the fastText baseline model, Embeddings from Language Models (ELMo) with a Support Vector Machine (SVM) classifier, and Bidirectional Encoder Representations from Transformers (BERT) as part of the classification process for requirements. To train the models, the authors created the PURE dataset, consisting of requirement and non-requirement text from 80 requirement documents. Based on their training results, they were able to achieve an F1-score of 0.86, a precision of 0.92, and a recall of 0.80 among classes for the BERT model. They found that BERT performed better for F1 and precision, while the other models performed better for recall [14]. Another study utilized a variety of BERT variants such as Scientific BERT (SciBERT), Robustly Optimized BERT Approach (RoBERTa), and Explainable Robust BERT (XRBERT). Additionally, the study utilized a variety of pre-processing techniques including stop-word removal, Part-of-Speech (POS) tagging, lemmatization, and Principal Component Analysis (PCA). Because of their limited dataset, they also utilized few-shot learning to help train their associated models. Although the few-shot learning implementations achieved impressive results, they were outperformed by other model training methods. Additionally, extensive data pre-processing did not drastically impact the result, while more simplistic models performed similarly to the BERT variants [24]. Lastly, another study utilized LLMs such as BERT, Distilled BERT (DistilBERT), Distilled RoBERTa (DistilRoBERTa), Efficiently Learning an Encoder that Classifies Token Replacements Accurately (Eltract-base), and Extra Long Net (XLNet) to perform classification of non-functional requirements. When compared to simplistic models such as Logistic Regression, SVM, and CNNs, they found that XLNet performed much better for NFR classification [25].

3.2 Requirement quality analysis

Existing requirement quality tools utilize different implementations and quality models to analyze requirements. As mentioned previously, the IEEE 29148 standard provides a quality model that the ARQM tool is primary based off. However, author studies and implementations utilize different quality models, even though the models typically measure the same quality deficiencies within requirement documents. A seminal article discussed a quality attribute model that consisted of eighteen primary measurements that correlate to their associated attributes. The article suggests that many of the quality attributes require human inspection and does not provide methods of automation [10]. Other implementations attempt to automate various parts of that quality model via the NASA ARM and reconstructed tools, finding textual statistics and indicators that can help measure requirement quality [1], [9]. Another study implemented a tool called *smella* to extend upon the implementation of text-based statistics. They utilize dictionaries and NLP-based methods to help detect violations such as ambiguous adverbs, ambiguous adjectives, and superlatives, improving the generalizability of heuristics [26], [27]. Lastly, a systematic study looked at multiple implementations of requirement smells among morophological, lexical, analytical, and relational smells. The study found that NLP and Machine Learning were popular for requirement smells but did not adequately utilize NLP in conjunction with ML to help uncover requirement violations. Additionally, lexical analysis was also the most popular, suggesting improvement potential in complexity of requirement smell detection [28].

Other existing rule-based practical tools include applications such as *SREE*, *RQA*, and *LELIE*. The SREE application is a rule-based tool that utilizes user-defined dictionaries to helps to improve the dynamic results of the application; the tool comes with default words and phrases that are adjustable for the user, allowing for tool adjustment [29]. The *RQA* tool focuses on step functions for heuristics. These step functions can positively or negatively correlate to quality attribute violations, allowing for a dynamic understanding of how textual representations impact quality violations in both a positive and negative manner [30]. The *LELIE* tool utilizes heuristics to identify complicated analytical errors, including semantical and discourse violations. The tool also allows for user feedback to help accommodate domain-specific requirement knowledge [31].

Other existing tools utilize NLP and AI-based strategies to help determine requirement quality. As an example *RETA* analyzes requirement template conformance through NLP strategies; the application simplifies the requirement text into a defined template to help determine what aspects of the requirement may be missing [32]. Another tool called *Detect Nominalizations* performs a similar function to determine requirement template conformance [33]. Other tools utilize alternative NLP and AI-based methods to analyze requirements. For instance, the *Nocuous Ambiguity Identification* tool utilizes ML to help identify potentially harmful ambiguities. The analysis goes beyond surface-level heuristics, while helping to uncover ambiguities that are likely to harm the requirement understanding [34]. Another tool called *ReqSAC* utilizes tree-based ML implementations to uncover Ambiguity from requirements. For the analysis of the ML-based tool, the results indicated a high agreement with the human-annotated dataset used for training.

Based on multiple systematic literature reviews, there are existing tools that screen, identify, and analyze requirements against associated quality. Although the tools utilize a wide range of methods, the techniques primarily do not use modern AI and NLP methods for automated requirement processing. As mentioned in the literature reviews, many of the tools are no longer available or actively supported, leaving a wide research gap in the utilization of modern methods for automated requirement processing. Table 1 Automated Requirement Methods and Tools Table 1 lists the various tools and methods available for requirement identification and analysis [3], [5].

Table 1 Automated Requirement Methods and Tools

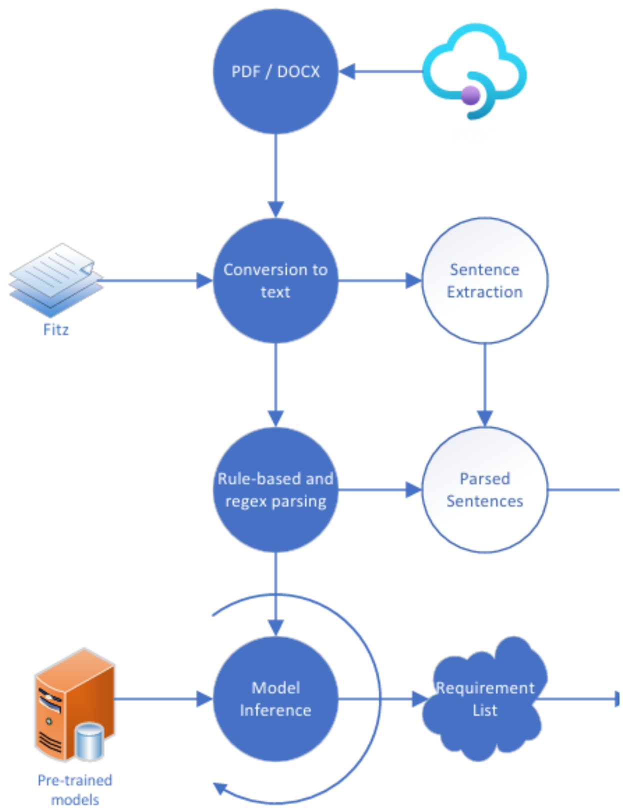
Requirement Identification and Screening	**Requirements-Collector** [35], **Requirement Mining Framework** [36], **Data Mining for RE** [37], **Graph-Based Elicitation** [38], **GOORE** [39], **Ontology-Guided Elicitation** [40], **JIT Requirements Capturer** [41], **Requirements Recommender** [42], **Mobile App Security RE** [43]
Quality Analysis	**Inconsistency Checker** [44], **Inconsistency Handling Assistant** [45], **Duplicate Use Case Detector** [46], **Linguistic Merge Tool** [5](Antinyan & Staron, 2017)**, RENDEX** [47]**, CIRCE** [48], **Ambiguity Detector** [49]**,** EA-Analyzer [50], **REInDetector** [51]**, MaramaAIC** [44]**, Spec Review & Test** [52]**, Quality User Story** [53]**, Requirements Quality Classifier** [54]**, Aspect Mining for Quality Attributes** [46]

The practical tools available for identification primarily utilize and analyze unstructured data. Most of the tools consolidate the unstructured format of requirement data into a structured form to help encapsulate potential requirements. These applications utilize the logical formulations of the structured data to help derive potential requirements; often these applications allow for the parsing of different types of input data, including audio transcription and external documentation. Similarly, the quality analysis tools try to consolidate the requirement documents into a formal representation before processing. Once the data is represented formally, the automated tools utilize methods such as semantic similarity to detect duplicate requirements. Additionally, these tools formulate rule-based algorithms for formal representation to check for inconsistencies and ambiguities. Ultimately, most tools utilize AI as a preliminary processing step before actually analyzing the requirement quality, limiting the scope of their analysis.

4. ARQM: An Automated Tool for Requirements Screening and Quality Analysis

The ARQM application seeks to expand upon existing implementations of requirement automation tools, while also utilizing modern AI and NLP methods to accomplish requirement screening and quality analysis. The tool's scope includes the ingestion of requirement documentation and the generation of an artifact that outlines the associated quality of the requirements. The main focus of the application was to simplify the user involvement, while providing useful feedback for requirement writing violations through the use of XAI. Additionally, the research looks at ways to separate and delineate requirements from unstructured requirement documents to allow for accurate processing. An overview of the general ARQM document ingestion process is provided in **Error! Reference source not found.**.

,



The current iteration of the tool evaluates select quality attributes for requirements and provides visual feedback for potential violation resolution. Additionally, the ARQM tool provides deep analysis of individual requirements using ML and transfer learning, allowing for lexical, syntactical, semantical, and pragmatic analysis. The results of the model are synthesized into a visual representation instead of individual class indicators, simplifying the training process (sample results are shown in **Error! Reference source not found.** and **Error! Reference source not found.**). Additionally, the tool utilizes various practical datasets that were either publicly available or manually annotated, increasing the generalizability of the requirement automation task.

Through the use of NLP and ML, the tool tries to utilize different techniques to achieve requirement identification and quality analysis. The application utilized multiple preprocessing strategies and models to achieve requirement identification, each with varying degrees of complexity and computational requirements. The tool also synthesizes the different types of requirements, such as functional and non-functional requirements, into a single category, allowing for only two classification outcomes. Primarily, the goal for requirement identification was to learn the complicated representations of requirements without the use of heuristics and pre-defined lists.

Additionally, the ARQM tool analyzes the quality attributes of *Singularity*, *Feasibility*, *Verifiability*, and *Ambiguity*. Through transfer learning, the tool learned the complex representations that could potentially indicate quality attribute violations from human annotators. Similar to requirement screening, the tool does not utilize heuristics. Instead, the tool utilizes transfer learning for various LLMs to provide general feedback based on requirement context. The learned knowledge of the LLM ideally transfers to the quality classification outcome, improving the scope and complexity of the analysis. As a result, the different types of quality analysis methods were synthesized as contributions to the model, limiting the scope of classification. Primarily, the classification feedback is shown through the contributions of textual units to the classification result, allowing for the visualizations of word and phrase contributions without the use of heuristics.

5. Validation and Evaluation

For validation, the focus was on both requirement identification and quality analysis. Through the utilization of both publicly available and human-annotated datasets, the process compares to the typical AI metrics such as accuracy, precision, recall and F1. For requirement identification, classification tasks were simpler; therefore, the discussion talks about the common heuristics in the form of words that often appeared during the model inference. These techniques either utilized the components of statistical models to visualize importance of individual words or synthesized more complicated models to statistical models to help extract out heuristics in the inference process.

For requirement screening, a variety of methods and models were utilized during training. Simpler models such as Logistic Regression and Support Vector Machine (SVM) helped encapsulate the general patterns of non-requirement and requirement data, while more complicated models such as RNNs and CNNs helped learn highly complicated representations of the data for better generalization. Primarily, these models were trained off of the PURE dataset, which was representative of a diverse amount of requirement documents. Because of the large dataset availability for requirement identification, the task showed consistent performance across models, suggesting the effectiveness of models for learning requirement and non-requirement representations.

For the analysis of quality attributes, the process utilized a human-annotated dataset for model training. Ultimately, multiple models were utilized to help classify the various quality attribute violations among a requirement. These models varied in complexity, although the analysis likely required a complicated understanding of violations and the associated context. As mentioned previously, all analysis types were synthesized into the model analysis, with a particular focus on pragmatic analysis; the likely reading and impact of a requirement is more important than simple heuristics. Still, the analysis includes the potential heuristics that often affect the outcome of the classification for each quality class, mitigating the need for complex models. The analysis also includes the rater agreement and key takeaways of practical human annotation with quality analysis.

For the PURE dataset, there were a total of 2833 requirements and 2475 non-requirement entries [7]. During the training process, we utilize the Synthetic Minority Over-sampling Technique (SMOTE) for the minority class to address dataset disparities. Similarly for the reviewer datasets, the training set had a proportion of 222 requirements and 198 non-requirements based on user input. For Ambiguity, the proportion was 162 Ambiguous requirements and 258 non-Ambiguous requirements. Additionally for Feasibility, the proportion was 407 Feasible requirements and 13 non-Feasible requirements. Notably, the dataset was not balanced for Feasibility, creating challenges with model training. For Singularity, the proportion was 321 Singular requirements and 99 non-Singular requirements and Verifiability had a proportion of 290 Verifiable requirements and 130 non-Verifiable requirements. Due to the high disparity of class representation for Feasibility, the quality attribute has been mostly eliminated from the analysis. The other quality attributes utilized SMOTE to help balance the datasets.

5.1. Requirement Screening Results

As mentioned before, the analysis includes a variety of different models and implementations to achieve accurate inference for the PURE dataset. Of these techniques, the methods included data pre-processing techniques, including methods such as stop-word removal and lemmatization. Critically, the analysis looks at the results of these preprocessing techniques, including their removal. Although requirement identification typically requires the understanding of textual context, preprocessing techniques can help to eliminate the complexity of the training data, allowing for the uncovering of more generic patterns. Table 2 shows the results across the various different models, organized by class.

Based on the results, DistilBERT outperformed almost every model, while also performing consistently high among both classes. For the negative class, DistilBERT outperformed on all metrics, while outperforming on Precision, F1, and Accuracy for the positive class. When the LLMs are eliminated, CNNs showed the most consistently high results among metrics.

Fig. **1** and Fig. **2** show the visuals of the metric results across models. Importantly, these visualizations showcase the results without the inclusion of LLMs such as DistilBERT. For both classes, KNN underperformed across most metrics. Complicated Deep Learning (DL) models outperformed for most metrics across the classification classes, with the CNN and RNN models performing the best among most metrics and classes. Other models such as Random Forest and the ensemble implementations provided consistently high results among both classes as well.

Simpler models such as Logistic Regression showed accurate results as well, suggesting that added model complexity does not necessarily help with broad requirement identification. Still, it seems that the CNN and RNN implementations outperformed the other models, due to their ability to analyze sequential data and the associated relationships. Ultimately, no models were able to beat the performance of DistilBERT.

Table 3 Illustrates the impact of lemmatization on the dataset results. For the table, an indicator with class 1, such as P_1, represents results before applying the transformation (such as lemmatization or pre-processing) and the second class represents the results with the transformation. Although there was partial improvement for some of the models, there were notable decreases through most of the model results. Additionally, this method of pre-processing was only useful for certain models due to the requirement of context for other LLM models including DistilBERT. The improvement among metrics is shown through bold casing. Notice, the improvements were also minimal, suggesting lemmatization did not significantly help with improving inference among models.

For the table, an indicator with class 1, such as P_1, represents results before applying the transformation (such as lemmatization or pre-processing) and the second class represents the results with the transformation. Although there was partial improvement for some of the models, there were notable decreases through most of the model results. Additionally, this method of pre-processing was only useful for certain models due to the requirement of context for other LLM models including DistilBERT. The improvement among metrics is shown through bold casing. Notice, the improvements were also minimal, suggesting lemmatization did not significantly help with improving inference among models.

Table 2 Requirement Identification Results with Stop Word Removal

Model	Class	Precision	Recall	F1-Score	Accuracy
ANN	N	0.81	0.86	0.84	0.85
CNN	N	0.82	0.93	0.87	0.87
XLNet	N	0.92	0.96	0.94	0.94
DistilBERT	N	**0.93**	**0.97**	**0.95**	**0.95**
DistilBERT Few-Shot	N	0.81	0.83	0.82	0.83
DistilBERT Siamese	N	0.92	0.77	0.84	0.86
Ensemble (Average)	N	0.81	0.86	0.83	0.84
Ensemble (Majority)	N	0.81	0.87	0.84	0.85
KNN	N	0.64	0.88	0.74	0.72
Logistic Regression	N	0.80	0.87	0.83	0.84
Meta-Model (Gradient Boost)	N	0.83	0.82	0.82	0.84
Meta-Model (Logistic Regression)	N	0.81	0.83	0.82	0.84
Random Forest	N	0.82	0.89	0.86	0.86
RNN	N	0.88	0.85	0.88	0.87
Support Vector Machine	N	0.80	0.89	0.84	0.85
XLNet	Y	0.96	0.92	0.94	0.94
ANN	Y	0.88	0.84	0.86	0.85
CNN	Y	0.93	0.82	0.87	0.87
DistilBERT	Y	**0.97**	0.93	**0.95**	**0.95**
DistilBERT Few-Shot	Y	0.85	0.83	0.84	0.83
DistilBERT Siamese	Y	0.82	**0.94**	0.88	0.86
Ensemble (Average)	Y	0.88	0.83	0.85	0.84
Ensemble (Majority)	Y	0.89	0.83	0.86	0.85
KNN	Y	0.86	0.59	0.70	0.72
Logistic Regression	Y	0.88	0.82	0.85	0.84
Meta-Model (Gradient Boost)	Y	0.85	0.86	0.86	0.84
Meta-Model (Logistic Regression)	Y	0.86	0.84	0.85	0.84
Random Forest	Y	0.90	0.84	0.87	0.86
RNN	Y	0.87	0.90	0.88	0.87
Support Vector Machine	Y	0.90	0.81	0.85	0.85

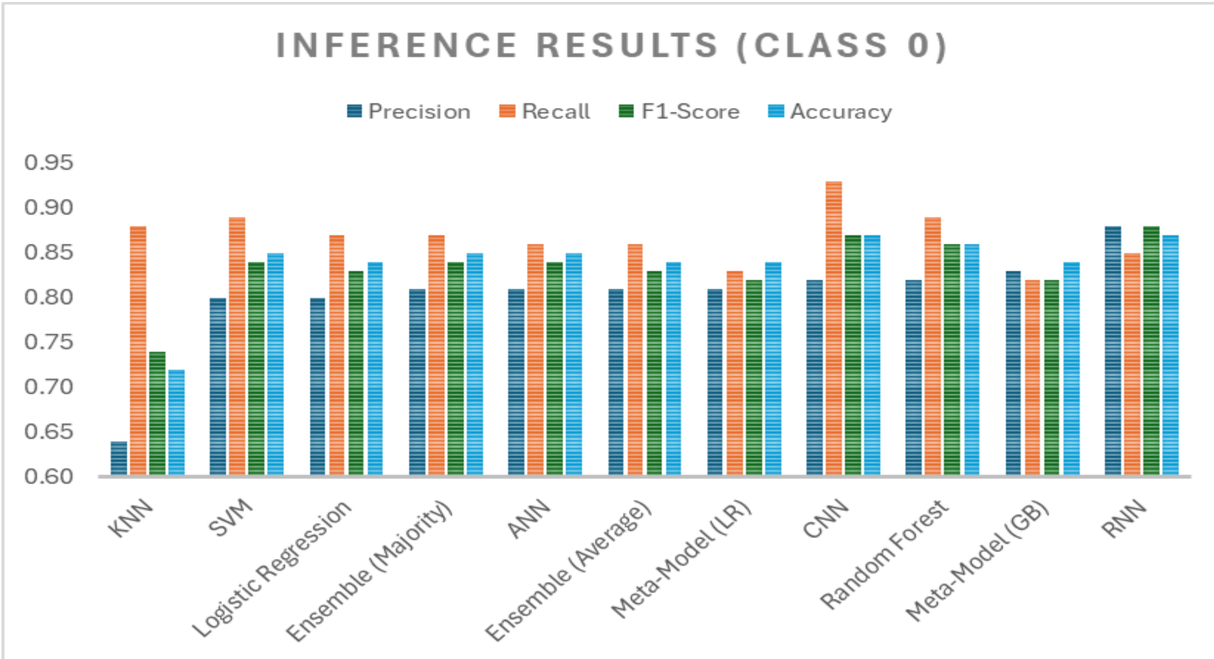


Fig. 1 *Inference Results (Class 0)*

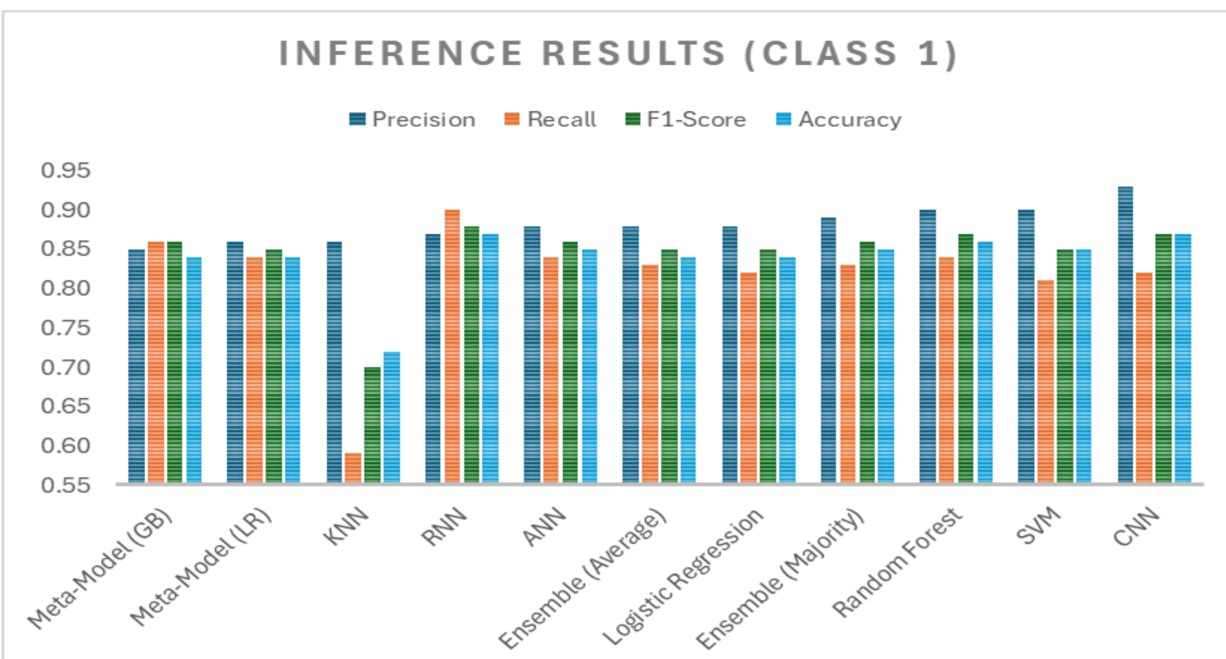


Fig. 2 *Inference Results (Class 1)*

Table 3 Requirement Identification Results with Lemmatization

Model	Class	P_1	P_2	R_1	R_2	F1_1	F1_2	A_1	A_2
ANN	N	0.81	0.79	**0.86**	**0.91**	0.84	0.84	0.85	0.85
KNN	N	0.64	0.63	**0.88**	**0.90**	0.74	0.74	0.72	0.72
Logistic Regression	N	0.80	0.79	0.87	0.86	0.83	0.82	0.84	0.83
Random Forest	N	0.82	0.80	0.89	0.88	0.86	0.84	0.86	0.85
Support Vector Machine	N	0.80	0.80	0.89	0.88	0.84	0.84	0.85	0.85
ANN	Y	**0.88**	**0.91**	0.84	0.80	0.86	0.85	0.85	0.85
KNN	Y	**0.86**	**0.87**	0.59	0.57	0.70	0.69	0.72	0.72
Logistic Regression	Y	0.88	0.87	0.82	0.82	0.85	0.84	0.84	0.83
Random Forest	Y	0.90	0.89	0.84	0.82	0.87	0.86	0.86	0.85
Support Vector Machine	Y	0.90	0.89	**0.81**	**0.82**	**0.85**	**0.86**	0.85	0.85

Table 4 discusses the results of the models that utilize a simple Count Vectorizer preprocessing technique. Surprisingly, there were minor improvements across most models and measurements, suggesting that pre-processing techniques did not significantly improve models, and that the sequential nature of the sentences mattered when considering the inference results. Similarly to previous results, statistical models were also able to achieve highly accurate inference results among both classes, indicating potential simplicity of requirement inference. For accuracy, Random Forest performed best with 0.87 for both classes; Random Forest also outperformed for F1 score, negative class recall, and positive class recall. Consistently high results are shown through the ANN model relative to alternative models.

Table 4 Requirement Identification Results without Preprocessing

Model	Class	P_1	P_2	R_1	R_2	F1_1	F1_2	A_1	A_2
ANN	N	0.79	0.81	0.91	0.91	0.84	0.86	0.85	0.86
KNN	N	0.63	0.61	0.90	0.95	0.74	0.75	0.72	0.71
Logistic Regression	N	0.79	0.80	0.86	0.86	0.82	0.83	0.83	0.84
Random Forest	N	0.80	0.81	0.88	0.93	0.84	0.86	0.85	0.87
Support Vector Machine	N	0.80	0.81	0.88	0.89	0.84	0.85	0.85	0.86
ANN	Y	0.91	0.92	0.80	0.83	0.85	0.87	0.85	0.86
KNN	Y	0.87	0.93	0.57	0.51	0.69	0.66	0.72	0.71
Logistic Regression	Y	0.87	0.88	0.82	0.82	0.84	0.85	0.83	0.84
Random Forest	Y	0.89	0.93	0.82	0.82	0.86	0.87	0.85	0.87
Support Vector Machine	Y	0.89	0.90	0.82	0.83	0.86	0.86	0.85	0.86

Lastly, Table 5 shows the results of the utilization of embeddings when compared to no preprocessing. These embeddings were generated with *Word2Vec* and were averaged out to formulate sentence-level embeddings during the inference process. The results did not indicate a major improvement among the metrics, but did show the elimination of extremities, such as recall for the KNN model. Notably, some metrics experienced lower performance amongst different models. The key model that achieved higher performance with embeddings was KNN, with minor limitations. Overall, it does not seem that embeddings necessarily improved classification performance for requirement identification significantly among statistical models.

Based on the synthesis of the model results, **Fig. 3** and **Fig. 4** show the most impactful words for model inference. For the negative class, words such as ‘administration’ , ‘processes’, ‘xml’, and ‘document’ were negatively indicative of a requirement classification, suggesting that non-requirements often contain these specific words or a specific sequence of them. Words such as ‘shall’, ‘should’, ‘must’, and ‘able’ ultimately contributed positively to requirement language, suggesting requirements typically contain more formal language. Intuitively, the results suggest that strict and rigid language typically correlates to requirements, whereas references and informal language are typically non-requirement related. Additionally, it also seems that the sequential nature of text requires word combinations for adequate classification instead of individual words, given the partial overlap of terminology between the two classes.

For requirement screening, the key differences between requirement and non-requirement text were not apparent, even with model training. Given the results of the various models on the PURE dataset, there is clearly overlap between the different classes.

Table 5 Requirement Identification Results with Embeddings

Model	Class	P_1	P_2	R_1	R_2	F1_1	F1_2	A_1	A_2
ANN	N	0.81	0.81	0.91	0.86	0.86	0.83	0.86	0.83
KNN	N	**0.61**	**0.79**	0.95	0.77	**0.75**	**0.78**	**0.71**	**0.79**
Logistic Regression	N	0.80	0.75	0.86	0.81	0.83	0.78	0.84	0.78
Random Forest	N	0.81	0.79	0.93	0.86	0.86	0.82	0.87	0.82
Support Vector Machine	N	0.81	0.75	0.89	0.83	0.85	0.79	0.86	0.78
ANN	Y	0.92	0.86	0.83	0.80	0.87	0.83	0.86	0.83
KNN	Y	0.93	0.79	**0.51**	**0.81**	**0.66**	**0.80**	**0.71**	**0.79**
Logistic Regression	Y	0.88	0.80	0.82	0.74	0.85	0.77	0.84	0.78
Random Forest	Y	0.93	0.85	0.82	0.78	0.87	0.81	0.87	0.82
Support Vector Machine	Y	0.90	0.82	0.83	0.74	0.86	0.78	0.86	0.78

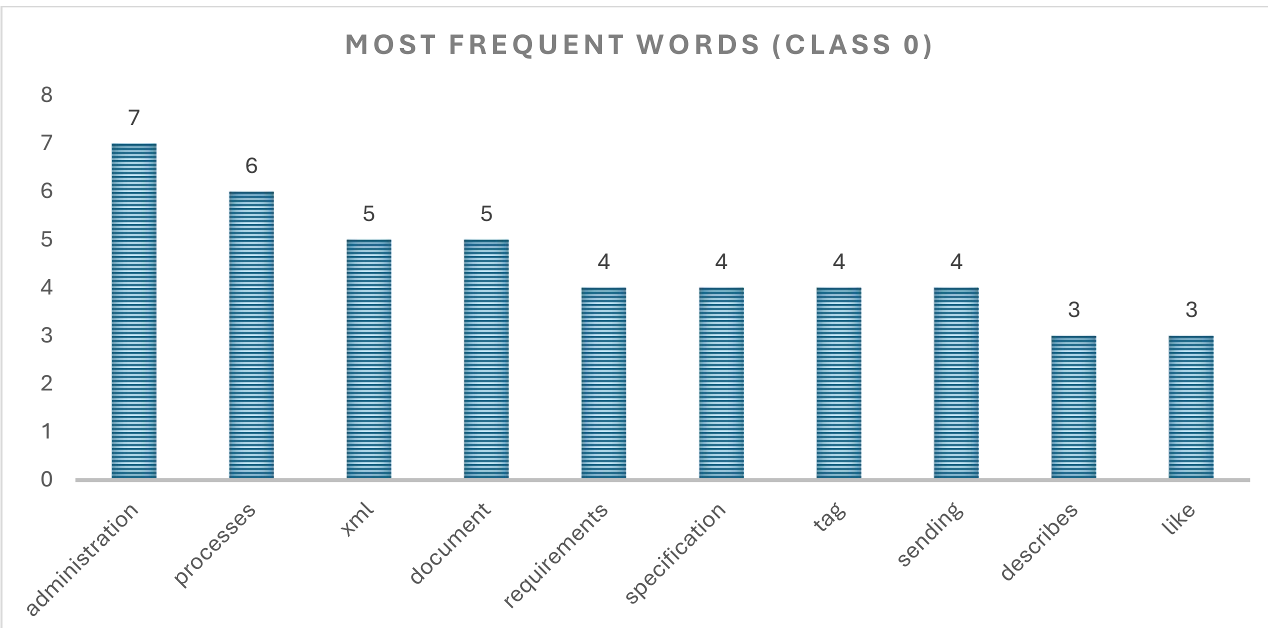


Fig. 3 *Top Words for the Negative Class*

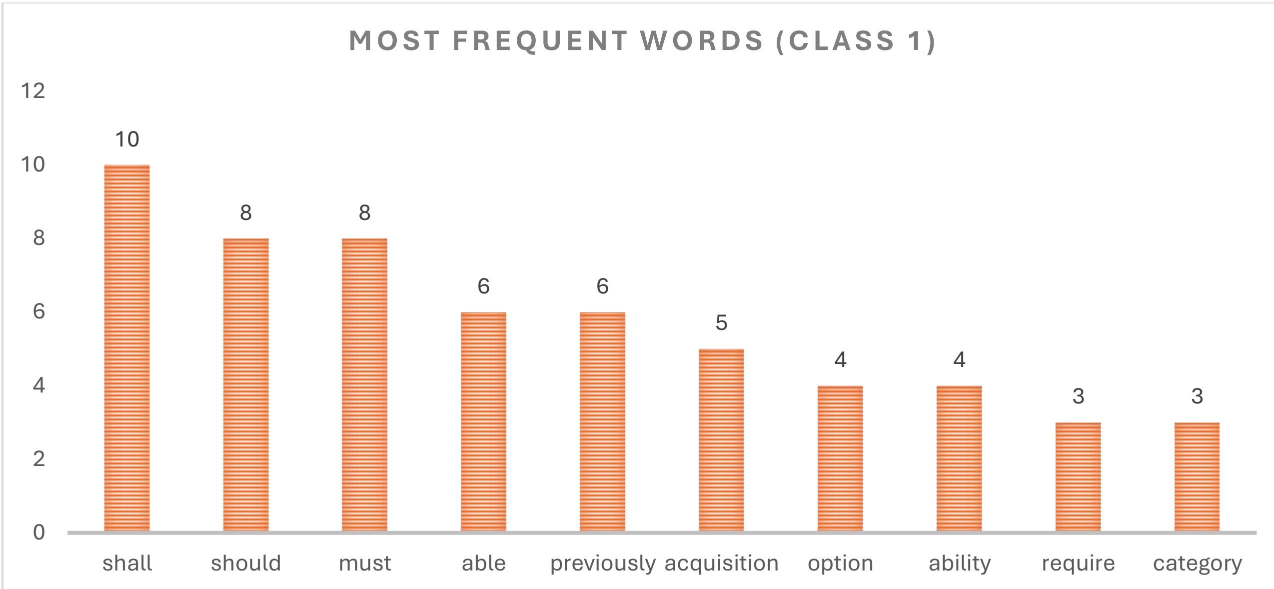


Fig. 4 Top Words for the Positive Class

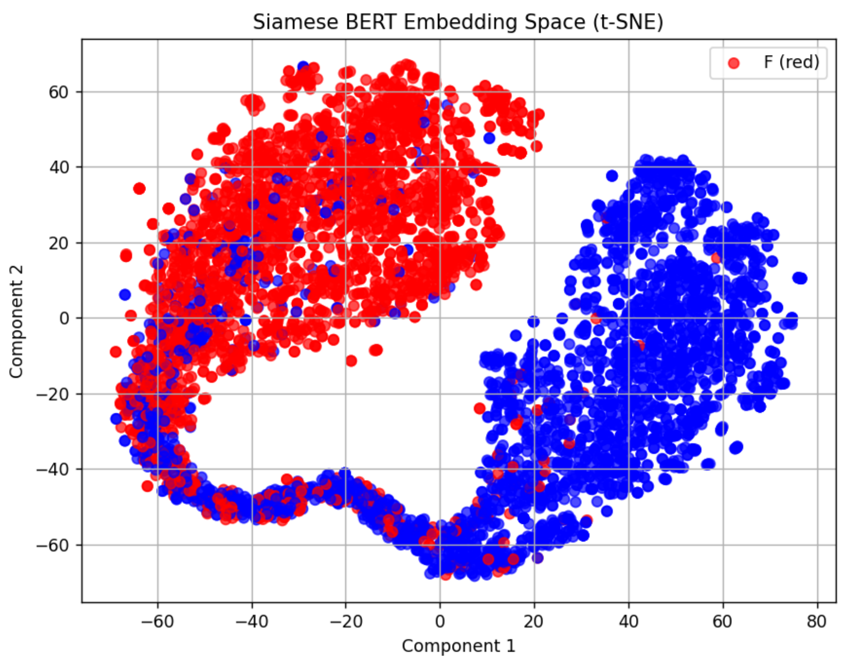


Fig. 5 T-SNE Visualization of Requirement classifications

In **Error! Reference source not found.Error! Reference source not found.**, the analysis shows the result of a Siamese BERT model; this model attempts to create embeddings that discriminate amongst the two classes. For instance, if the input contains two divergent examples, the cosine similarity is maximized. From this model, we synthesize the features that allow for class delineation through dimensionality reduction. **Error!**

Reference source not found. shows that although it is easy to distinguish between the two classes, there is a significant amount of overlap, limiting the applicability of the results. Regardless, the results show that generally, identifying requirement-related text is possible through both complicated and simplistic models, allowing for accurate and efficient automation of requirement identification

5.2. Human Annotated Screening and Quality Analysis Results

For requirement quality analysis and requirement screening, the models for ARQM were primarily trained on human-annotated datasets (in addition to the utilization of the PURE dataset). For the analysis process, the models analyzed the results for ambiguity, feasibility, verifiability, and singularity, quality attributes provided by the IEEE 29148 standard. The human annotators were given sets of requirements and were instructed to label them based on the definitions of the standard. Importantly, the datasets were overlapped so that multiple annotators labeled the same requirements. As a result, the study analyzes the rater agreement and overlap of the annotated datasets, providing insights into the reliability and consistency of human quality perception. Table 6 shows the agreement from the manual annotation with the associated PURE results. Table 7 shows the Kapa agreement among reviewers for requirement identification.

Table 6 Human Agreement with PURE

Dataset 1	0.286
Dataset 2_1	0.343
Dataset 2_2	0.286
Dataset 3_1	0.429
Dataset 3_2	0.4
Dataset 4	0.371
Dataset 5_1	0.371
Dataset 5_2	0.457
Dataset 6_1	0.457

Table 7 Annotator Kappa Agreement

	Annotator_1	*Annotator_2*	*Annotator_3*	*Annotator_4*	*Annotator_5*
Annotator_1	X	0.169	X	X	X
Annotator_2	0.169	X	0.441	X	X
Annotator_3	X	0.441	X	X	X
Annotator_4	X	X	X	X	0.6
Annotator_5	X	X	X	0.6	X

Based on the results outlined in Table 6 and Table 7, there was moderate agreement between the PURE dataset and the human annotated dataset. Additionally, there was moderate Kappa agreement among reviewers, suggesting a shared agreement for what constitutes a requirement. Table 8 shows the metrics achieved among the various models for the human annotated dataset. XLNet outperformed for accuracy with a score of 0.93. Additionally, XLNet performed better with non-requirement identification with recall and F1 scores of 0.92 and 0.93, respectively, while the Siamese Distilbert model outperformed for recall and F1 for the positive class. Additionally, other models such as the logistic meta-model, and ensembles achieved an accuracy of 0.90.

Next, the analysis shows the agreement among the annotators for the various requirement quality attributes. Table 9 shows the Kappa agreement among reviewers for the different quality attributes. Among reviewers 2, 3, 4, and 5, there was significant agreement for what constituted an ambiguity violation based on overlapped datasets. The table also showed moderate agreement among reviewers for Feasibility violations. Similarly, the results showed limited agreement but seemingly indicated less of a consistent pattern for singularity. Lastly, the table showed minor agreement for verifiability; based on the Kappa results, singularity and verifiability had minimal agreement among reviewers, whereas ambiguity and feasibility had high agreement.

In addition to Kappa agreement, the analysis looks at the ability of models to learn the various quality attributes based on the annotated dataset. Shown in Table 10, Table 11, and Table 12 are the associated results of the models for inference among the human annotated datasets. Because the feasibility category primarily contained positive classifications, the quality attribute is excluded from the results. Primarily, the models were able to achieve high accuracy among the various quality attributes. The highest accuracy for ambiguity was achieved by Siamese BERT with an accuracy of 0.85. For singularity, Siamese BERT also outperformed with an accuracy of 0.92. Lastly, the ANN implementation achieved an accuracy of 0.89 for Verifiability. **Error! Reference source not found.** visualizes the separation of classes among quality attributes.

Fig. 6 shows the embedding separations of the various quality attributes. For ambiguity and verifiability, there was minor separation and lack of agreement for what constituted a violation. Alternatively, feasibility and singularity had high separation among both classes, showing a clear separation of features for violations and non-violations for requirements. Ultimately, all quality attributes showed a degree of separation, suggesting some agreement among the human annotated dataset as to what constituted quality violations. These results may also suggest more complicated interpretations for Ambiguity and Verifiability, as the other quality attributes may have more simplistic definitions among reviewers [7]. This dataset presents PURE (PUblic REquirements dataset), a dataset of 79 publicly available natural language requirements documents collected from the Web. The dataset includes 34,268 sentences and can be used for natural language

Table 8 Model Results for Annotated Identification

Model	***Class***	***Precision***	***Recall***	***F1-Score***	***Accuracy***
ANN	N	0.83	0.82	0.83	0.83
CNN + Embeddings	N	0.90	0.81	0.85	0.86
XLNet	N	**0.94**	**0.92**	**0.93**	**0.93**
DistilBERT	N	0.83	0.89	0.86	0.84
DistilBERT Few-Shot	N	0.74	0.91	0.82	0.79
DistilBERT Siamese	N	**1.00**	0.85	**0.92**	0.92
Ensemble (Average)	N	0.90	0.91	0.90	0.90
Ensemble (Majority)	N	0.88	0.91	0.90	0.90
KNN	N	0.68	0.94	0.79	0.75
Logistic Regression	N	0.87	0.91	0.89	0.89
Meta-Model (Gradient Boost)	N	0.92	0.81	0.86	0.87
Meta-Model (Logistic Regression)	N	0.91	0.90	0.90	0.90
Random Forest	N	0.86	0.93	0.89	0.89
RNN	N	0.83	0.84	0.84	0.84
Support Vector Machine	N	0.88	0.88	0.88	0.88
ANN	Y	0.82	0.84	0.83	0.83
CNN + Embeddings	Y	0.82	0.91	0.86	0.86
XLNet	Y	0.91	0.94	**0.92**	**0.93**
DistilBERT	Y	0.86	0.80	0.83	0.84
DistilBERT Few-Shot	Y	0.86	0.64	0.74	0.79
DistilBERT Siamese	Y	0.86	**1.00**	**0.92**	0.92
Ensemble (Average)	Y	0.91	0.90	0.90	0.90
Ensemble (Majority)	Y	0.91	0.88	0.89	0.90
KNN	Y	0.90	0.55	0.69	0.75
Logistic Regression	Y	0.91	0.87	0.89	0.89
Meta-Model (Gradient Boost)	Y	0.83	0.93	0.87	0.87
Meta-Model (Logistic Regression)	Y	0.90	0.91	0.90	0.90
Random Forest	Y	**0.92**	0.85	0.88	0.89
RNN	Y	0.86	0.84	0.85	0.84
Support Vector Machine	Y	0.88	0.88	0.88	0.88

Table 9 Comparison of Annotators on Requirement Quality Attributes

Ambiguity	-0.068	0.304	0.711
Feasibility	0	0.66	0.317
Singularity	0.333	0.151	0.204
Verifiability	0	0.172	0.117

**kappa: 0-0.2 Slight, 0.21-0.4 Fair, 0.41-0.6 Moderate, 0.61 - 0.8 Substantial, > 0.8 Almost Perfect*

processing tasks that are typical in requirements engineering, such as model synthesis, abstraction identification and document structure assessment. It can be further annotated to work as a benchmark for other tasks, such as ambiguity detection, requirements categorization and identification of equivalent requirements. In the associated paper, we present the dataset and we compare its language with generic English texts, showing the peculiarities of the requirements jargon, made of a restricted vocabulary of domain-specific acronyms and words, and long sentences. We also present the common XML format to which we have manually ported a subset of the documents, with the goal of facilitating replication of NLP experiments. The XML documents are also available for download [7].

Because of the moderate and low Kappa agreement, the analysis also looks at the model agreement. In particular, the analysis utilizes the Fleiss' Kappa, a metric that measures relative agreement among models. For requirement identification tasks, there was a high Fleiss' Kappa agreement, suggesting that patterns learned from models are relatively similar. Even though the Cohen's Kappa among datasets were limited, Fleiss' Kappa also shows strong and moderate agreement among requirement quality tasks, especially for singularity. Models tasked with verifying Ambiguity and Verifiability had moderate agreement for classification. In the analysis, the models analyzed for agreement included Logistic Regression, SVM, Random Forest, Naïve Bayes, KNN, ANN, CNN, and RNN.

Table 10 Model Results for Ambiguity

ANN	N	0.73	0.51	0.61	0.63
CNN + Embeddings	N	0.71	0.65	0.68	0.70
XLNet	N	0.83	0.78	0.80	0.80
DistilBERT	N	0.76	0.82	0.79	0.78
DistilBERT Few-Shot	N	0.60	**0.88**	0.71	0.59
DistilBERT Siamese	N	**0.88**	0.86	**0.87**	**0.85**
Ensemble (Average)	N	0.68	0.69	0.68	0.69
Ensemble (Majority)	N	0.67	0.75	0.71	0.70
KNN	N	0.57	**0.88**	0.69	0.63
Logistic Regression	N	0.69	0.71	0.70	0.70
Meta-Model (Gradient Boost)	N	0.69	0.61	0.65	0.68
Meta-Model (Logistic Regression)	N	0.71	0.69	0.70	0.72
Random Forest	N	0.82	0.77	0.79	0.78
RNN - LSTM + Embeddings	N	0.87	0.69	0.77	0.77
Support Vector Machine	N	0.81	0.63	0.71	0.72
ANN	Y	0.57	0.78	0.66	0.63
CNN + Embeddings	Y	0.68	0.74	0.71	0.70
XLNet	Y	**0.78**	0.83	0.80	0.80
DistilBERT	Y	0.80	0.74	0.77	0.78
DistilBERT Few-Shot	Y	0.50	0.17	0.26	0.59
DistilBERT Siamese	Y	0.79	0.81	0.80	**0.85**
Ensemble (Average)	Y	0.71	0.69	0.70	0.69
Ensemble (Majority)	Y	0.74	0.66	0.70	0.70
KNN	Y	**0.78**	0.39	0.52	0.63
Logistic Regression	Y	0.72	0.70	0.71	0.70
Meta-Model (Gradient Boost)	Y	0.67	0.74	0.70	0.68
Meta-Model (Logistic Regression)	Y	0.72	0.74	0.73	0.72
Random Forest	Y	0.74	0.79	0.77	0.78
RNN - LSTM + Embeddings	Y	0.69	**0.87**	0.77	0.77
Support Vector Machine	Y	0.65	0.83	0.73	0.72

Table 11 Model Results for Singularity

ANN	N	0.85	0.83	0.84	0.83
CNN + Embeddings	N	0.90	0.87	0.89	0.89
XLNet	N	**0.93**	0.77	0.84	0.87
DistilBERT	N	0.92	0.86	0.89	0.89
DistilBERT Few-Shot	N	0.73	**0.97**	0.83	0.71
DistilBERT Siamese	N	**0.93**	**0.97**	**0.95**	**0.92**
Ensemble (Average)	N	0.86	0.81	0.84	0.83
Ensemble (Majority)	N	0.88	0.83	0.86	0.85
KNN	N	0.73	0.88	0.80	0.77
Logistic Regression	N	0.90	0.82	0.86	0.86
Meta-Model (Gradient Boost)	N	0.90	0.65	0.76	0.78
Meta-Model (Logistic Regression)	N	0.90	0.77	0.83	0.83
Random Forest	N	**0.93**	0.89	0.91	0.91
RNN	N	0.90	0.78	0.83	0.84
Support Vector Machine	N	0.91	0.84	0.87	0.88
ANN	Y	0.82	0.84	0.83	0.83
CNN + Embeddings	Y	0.88	0.91	0.90	0.89
XLNet	Y	0.82	**0.95**	0.88	0.87
DistilBERT	Y	0.86	0.93	0.89	0.89
DistilBERT Few-Shot	Y	0.40	0.06	0.10	0.71
DistilBERT Siamese	Y	0.88	0.76	0.82	**0.92**
Ensemble (Average)	Y	0.81	0.86	0.83	0.83
Ensemble (Majority)	Y	0.83	0.88	0.85	0.85
KNN	Y	0.84	0.66	0.73	0.77
Logistic Regression	Y	0.82	0.90	0.86	0.86
Meta-Model (Gradient Boost)	Y	0.71	0.92	0.80	0.78
Meta-Model (Logistic Regression)	Y	0.79	0.90	0.84	0.83
Random Forest	Y	**0.89**	0.92	**0.91**	0.91
RNN	Y	0.79	0.90	0.85	0.84
Support Vector Machine	Y	0.84	0.91	0.88	0.88

Table 12 Model Results for Verifiability

ANN	N	0.86	**0.97**	**0.91**	**0.89**
CNN	N	0.91	0.72	0.80	0.82
XLNet	N	0.86	0.94	0.90	0.88
DistilBERT	N	0.80	0.90	0.85	0.84
DistilBERT Few-Shot	N	0.67	0.91	0.77	0.65
DistilBERT Siamese	N	**0.96**	0.86	0.90	0.87
Ensemble (Average)	N	0.81	0.66	0.73	0.72
Ensemble (Majority)	N	0.86	0.74	0.80	0.79
KNN	N	0.70	0.80	0.74	0.69
Logistic Regression	N	0.85	0.69	0.76	0.76
Meta-Model (Gradient Boost)	N	0.80	0.37	0.50	0.59
Meta-Model (Logistic Regression)	N	0.84	0.52	0.64	0.67
Random Forest	N	0.87	0.85	0.86	0.84
RNN	N	0.86	0.80	0.83	0.83
Support Vector Machine	N	0.86	0.67	0.75	0.75
ANN	Y	**0.95**	0.79	**0.86**	**0.89**
CNN	Y	0.76	**0.93**	0.84	0.82
XLNet	Y	0.91	0.80	0.85	0.88
DistilBERT	Y	0.88	0.78	0.83	0.84
DistilBERT Few-Shot	Y	0.50	0.16	0.24	0.65
DistilBERT Siamese	Y	0.74	0.92	0.82	0.87
Ensemble (Average)	Y	0.65	0.80	0.72	0.72
Ensemble (Majority)	Y	0.72	0.84	0.78	0.79
KNN	Y	0.68	0.55	0.61	0.69
Logistic Regression	Y	0.68	0.84	0.75	0.76
Meta-Model (Gradient Boost)	Y	0.52	0.88	0.65	0.59
Meta-Model (Logistic Regression)	Y	0.58	0.87	0.70	0.67
Random Forest	Y	0.81	0.84	0.83	0.84
RNN	Y	0.80	0.86	0.83	0.83
Support Vector Machine	Y	0.67	0.86	0.75	0.75

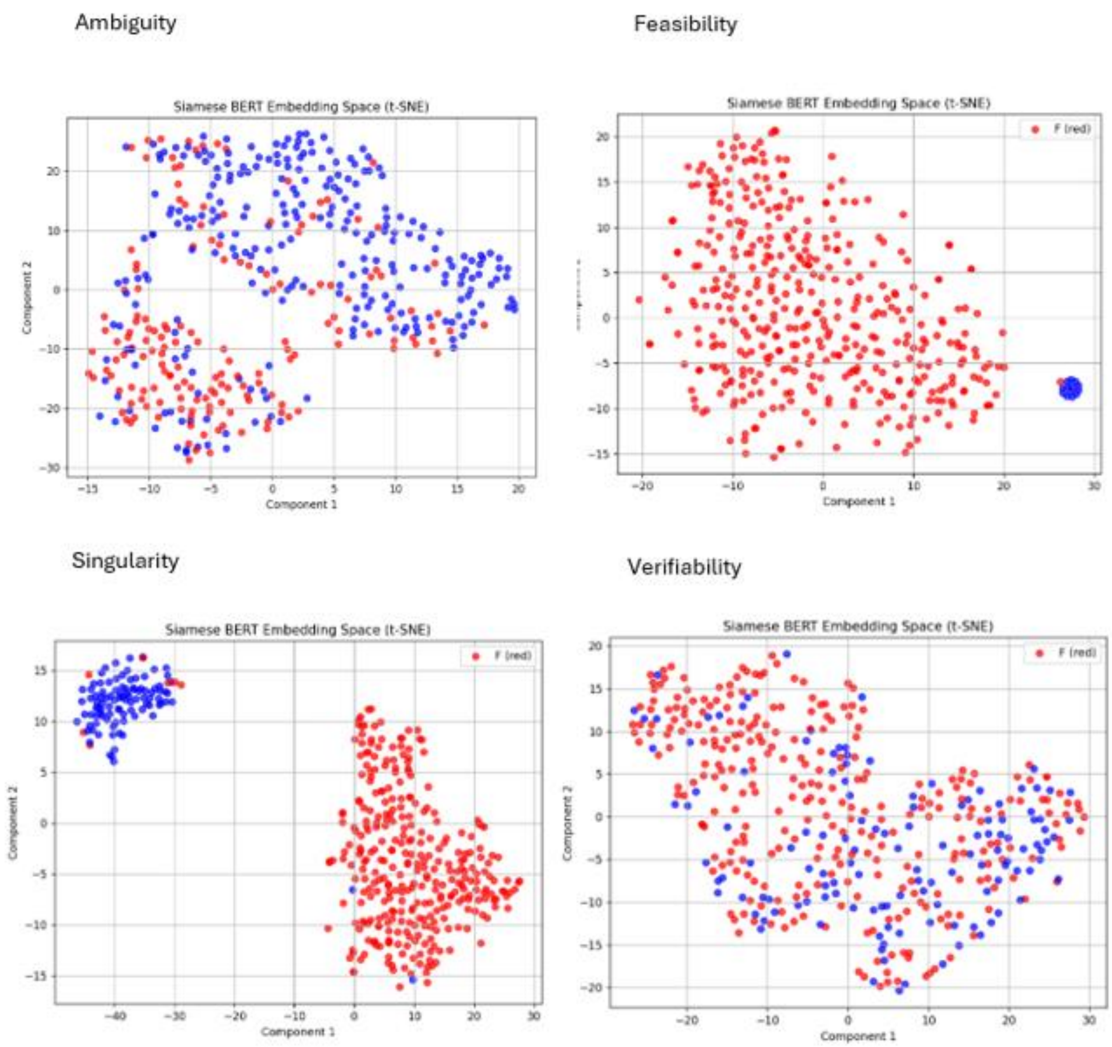


Fig. 6 *Embedding Space among Quality Attributes*

Table 13 Fleiss' Kappa Agreement

Method	*Fleiss' Kappa*
PURE Requirement Identification	0.749
Human Requirement Identification	0.703
Ambiguity	0.546
Singularity	0.641
Verifiability	0.542

6. Discussion

6.1. Requirement identification results

For the automated requirement identification experiment, the results showed moderate accuracy for the PURE dataset. Primarily, complicated models like DistilBERT outperformed for model inference among both classes, suggesting that requirement identification may require complicated LLMs to achieve higher accuracy. When removing DistilBERT from the analysis, CNN and RNNs performed the best among the associated metrics, suggesting an improvement in retaining critical contextual information for requirement identification. CNNs were generally outperformed by RNNs for both classes, although both models achieved the best results among simplistic implementations. Ultimately, DistilBERT performed significantly better than all models for both classes.

Surprisingly, pre-processing techniques did not make a significant positive difference for requirement identification among simpler models. The improvement among most models with no pre-processing suggests that pre-processing may hurt the accuracy of requirement identification. Simplistic, statistical-based models achieved relatively high accuracy among both classes without pre-processing as well, showing that there are clear separations of both classes for most instances of requirements and non-requirement text. When considering the accuracy of these simplistic models, the analysis utilized XAI and meta models to synthesize the most common word occurrences present within requirement and non-requirement text.

For requirements, words such as *shall, should,* and *must* were common among important model features. For non-requirements, words such as *administration*, *processes*, and *xml* were also common. With a manual inspection of the dataset, these words were present among both classes, suggesting that a combination of them impacted the results more than their individual contributions. Ultimately, formal language seemingly indicated requirement text whereas descriptive text indicated non-requirement text. With the reduction of high embedding vectors, there was still high overlap between the two classes, suggesting either limitations with the PURE dataset, or a more complicated dynamic for classification.

Although LLMs performed the best among classes, many models achieved adequate accuracy for most classification tasks. CNNs and RNNs, which are relatively lightweight DL models, achieved on-par accuracy with statistical models; their ability to capture sentence context proved useful for ambiguous cases of text. Additionally, simplistic, statistical-based models showed high precision, recall, and accuracy, further improving on the practical abilities of automated requirement identification through fast and general inference. Meta models showed stable results among classes, suggesting potential improvement for generalizability among requirement identification.

6.2. Requirement quality analysis results

The quality analysis showed impressive results for all quality attributes, achieving high metrics and accuracy for each classification. For Ambiguity, the Siamese BERT model achieved an accuracy of 0.85. For Singularity, the random forest model achieved an accuracy of 0.91. Lastly, the ANN model achieved an accuracy of 0.89 for the Verifiability quality attribute. Among all of the quality attributes, there were clear patterns and generalizations that models helped uncover, allowing for the utilization of AI for quality analysis. Additionally, many simplistic models achieved high metrics, limiting the need for complicated models and inference to achieve high accuracy among human annotated datasets.

For the manually annotated dataset for requirement identification, there was a high annotation agreement and result among models. For instance, Siamese BERT achieved an accuracy of 0.92, while simpler models achieved comparable accuracy. The Support Vector Machine, Random Forest, and Logistic Regression models achieved an accuracy of 0.88, 0.89, and 0.89, respectively. Additionally, various models achieved high precision, recall, and F1. The relatively balanced results among classes also indicate that the models learned well across the annotated dataset.

Lastly, the Kappa agreement was notable among both requirement identification and quality analysis tasks. For requirement identification, moderate Kappa agreement was achieved among reviewers with agreements of 0.169, 0.441, and 0.600. Additionally, there were high Kappa agreements for Ambiguity and Feasibility, but limited agreements for Singularity and Verifiability. Ultimately, the results were able to encapsulate the patterns indicated by the reviewers with moderate agreement. Because Feasibility did not have enough entries among both classes, the results were limited. Accordingly, the embedding space separations showed a strong separation of classes for some of the quality attribute determinations, such as Singularity and Feasibility, while showing limited embedding separation for Ambiguity and Verifiability.

The Fleiss' Kappa helped provide better insight into the applicability of models for determining quality violations. For Ambiguity, the models achieved a Fleiss' Kappa of 0.546, indicating a moderately strong agreement on which requirements are or are not ambiguous. Additionally, moderate agreement was also achieved for Singularity and Verifiability with Fleiss' Kappa scores of 0.641 and 0.542, respectively. Despite the low annotator agreement within the Verifiability attribute and in spite of the strong overlap with the embedding visualization, Fleiss' Kappa score of 0.542 indicates moderately strong model agreement regarding the verifiability of requirements.

6.3. Limitations

Of the existing results, the manually labeled datasets were limited. For instance, there was not an adequate representation of classes for the Feasibility quality attribute dataset, limiting the results and generalizability of models trained to detect violations. Additionally, although there was user agreement, there was also disagreement among each quality attribute for what constituted a quality violation. The limitations of Kappa agreement were not as impactful for requirement identification.

Another limitation was the data utilized by the reviewers. Although the study utilized real requirement documents, the documents were formulated in an academic setting and were created by students. Although the annotators had experience with RE, there were limitations that limited the overall generalization of the results. Additionally, the requirements were only derived from requirement documents; Often, requirements present themselves in multiple different formats. Therefore, the results are limited to certain requirement representations as well.

Lastly, the reviewers did not have additional context or important information related to the associated requirements. Because the text stood in isolation, the reviewers could not consider domain-related context, background, or project-specific restrictions before making a determination. This associated background data could likely have an impact on both requirement screening and requirement quality analysis, limiting the generalizability of the results as well. Although the results indicated a moderate and low Kappa agreement among various quality attributes, Fleiss' Kappa suggests that the models learned similar features during training.

7. Conclusion

In conclusion, we have created a practical tool named ARQM to help automate the requirement screening and requirement quality analysis process. This tool synthesizes modern AI techniques and human annotated datasets to achieve automation. Based on the results, AI models achieved accuracy for both automated tasks. Primarily, the tool analyzes four quality attributes in addition to requirement identification. Of the existing IEEE 29148:2018 quality attributes, the tool analysis ambiguity, feasibility, singularity, and verifiability.

With the results of the models, the tool utilizes XAI methodologies to visualize the most important tokens during inference. From these results, the user can see and understand the reasons behind potential quality violations. Overall, the tool focuses on simplistic and minimal user input. Primarily, the tool utilizes document ingestion, ideal AI models derived from our research, and a simple PDF generation process to help generate a quality report.

Once the quality report is generated, the user can visualize the individual requirements, quality attribute violations, and contributory text. Additionally, the ARQM system provides general metrics of document quality and total violations. ARQM builds upon past systems that tried to automate the RE process. Instead of relying on rule-based heuristics, ARQM utilizes the tacit knowledge present within modern AI models, allowing for more complicated and holistic inference.

Future work on the ARQM system will likely include an extension of quality attribute support, smarter inference, and better visualization. Additionally, the tool will likely focus on the utilization of summarization, and RAG implementations to help better improve inference. For better results, further research could create more extensive datasets that are broadly representative of requirement experts and end users. Although the tool has a large practical component, additional user acceptance testing would prove helpful.

Data Availability

Repositories:

1. https://github.com/burghporter314/ARQM-Finalized
2. https://github.com/burghporter314/Automated-Requirements-Quality-Measurement-Tool-Server

Datasets: To access the PURE dataset for requirement identification, the data is available at https://zenodo.org /records/1414117. To access the quality analysis data, please send a request to Dr. DeFranco (jfd104@psu.edu).

Statements and Declarations

Competing Interests: All authors have contributed to the concepts and formulations of the paper content. The paper largely relied on the NASA ARM tool and was inspired by a primary author of the tool, Dr. Phillip Laplante. Additionally, this paper was strongly adapted from Dylan Porter's unpublished praxis, which will eventually be submitted for consideration for relevant Pennsylvania State University publishing repositories. The content in this paper is not submitted to other journals for consideration.

References

[1] N. Carlson and P. Laplante, “The NASA automated requirements measurement tool: a reconstruction,” *Innov. Syst. Softw. Eng.*, vol. 10, no. 2, Art. no. 2, June 2014, doi: 10.1007/s11334-013-0225-8.

[2] P. S. Kummler, “Automated Quality Assessment of Natural Language Requirements.” Accessed: Sept. 16, 2023. [Online]. Available: https://publikationen.bibliothek.kit.edu/1000137191

[3] L. Zhao *et al.*, “Natural Language Processing for Requirements Engineering: A Systematic Mapping Study,” *ACM Comput. Surv.*, vol. 54, no. 3, Art. no. 3, Apr. 2021, doi: 10.1145/3444689.

[4] K. Kaur and P. Kaur, “The application of AI techniques in requirements classification: a systematic mapping,” *Artif. Intell. Rev.*, vol. 57, no. 3, Art. no. 3, Feb. 2024, doi: 10.1007/s10462-023-10667-1.

[5] M. A. Umar and K. Lano, “Advances in automated support for requirements engineering: a systematic literature review,” *Requir. Eng.*, vol. 29, no. 2, pp. 177–207, June 2024, doi: 10.1007/s00766-023-00411-0.

[6] “ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering,” *ISOIECIEEE 291482018E*, pp. 1–104, Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686.

[7] A. Ferrari, G. O. Spagnolo, and S. Gnesi, “PURE: a Dataset of Public Requirements Documents.” Zenodo, Sept. 12, 2018. Accessed: Sept. 09, 2025. [Online]. Available: https://zenodo.org/records/1414117

[8] P. A. Laplante and M. H. Kassab, *Requirements Engineering for Software and Systems*. Milton, UNITED KINGDOM: Auerbach Publishers, Incorporated, 2022. Accessed: Sept. 02, 2023. [Online]. Available: http://ebookcentral.proquest.com/lib/pensu/detail.action?docID=6927367

[9] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, “Automated analysis of requirement specifications,” in *Proceedings of the 19th international conference on Software engineering - ICSE ’97*, Boston, Massachusetts, United States: ACM Press, 1997, pp. 161–171. doi: 10.1145/253228.253258.

[10] A. Davis *et al.*, “Identifying and measuring quality in a software requirements specification,” in *[1993] Proceedings First International Software Metrics Symposium*, May 1993, pp. 141–152. doi: 10.1109/METRIC.1993.263792.

[11] P. M. Nugues, *Python for Natural Language Processing: Programming with NumPy, scikit-learn, Keras, and PyTorch*. in Cognitive Technologies. Cham: Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-57549-5.

[12] U. S. Shanthamallu and A. Spanias, *Machine and Deep Learning Algorithms and Applications*. in Synthesis Lectures on Signal Processing. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-031-03758-0.

[13] T. Clement, N. Kemmerzell, M. Abdelaal, and M. Amberg, “XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process,” *Mach. Learn. Knowl. Extr.*, vol. 5, no. 1, Art. no. 1, Mar. 2023, doi: 10.3390/make5010006.

[14] V. Ivanov, A. Sadovykh, A. Naumchev, A. Bagnato, and K. Yakovlev, “Extracting Software Requirements from Unstructured Documents,” in *Recent Trends in Analysis of Images, Social Networks and Texts*, E. Burnaev, D. I. Ignatov, S. Ivanov, M. Khachay, O. Koltsova, A. Kutuzov, S. O. Kuznetsov, N. Loukachevitch, A. Napoli, A. Panchenko, P. M. Pardalos, J. Saramäki, A. V.

Savchenko, E. Tsymbalov, and E. Tutubalina, Eds., Cham: Springer International Publishing, 2022, pp. 17–29. doi: 10.1007/978-3-031-15168-2_2.
[15] A. Ahmad *et al.*, "A Systematic Literature Review on Using Machine Learning Algorithms for Software Requirements Identification on Stack Overflow - Ahmad - 2020 - Security and Communication Networks - Wiley Online Library." Accessed: Oct. 24, 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/2020/8830683
[16] J. Jeon, X. Xu, Y. Zhang, L. Yang, and H. Cai, "Extraction of Construction Quality Requirements from Textual Specifications via Natural Language Processing," *Transp. Res. Rec.*, vol. 2675, no. 9, pp. 222–237, Sept. 2021, doi: 10.1177/03611981211001385.
[17] H. Akay, M. Yang, and S.-G. Kim, "Automating Design Requirement Extraction From Text With Deep Learning," in *Volume 3B: 47th Design Automation Conference (DAC)*, Virtual, Online: American Society of Mechanical Engineers, Aug. 2021, p. V03BT03A035. doi: 10.1115/DETC2021-66898.
[18] J. M. Calle Gallego and C. M. Zapata Jaramillo, "QUARE: towards a question-answering model for requirements elicitation," *Autom. Softw. Eng.*, vol. 30, no. 2, p. 25, July 2023, doi: 10.1007/s10515-023-00386-w.
[19] B. Kwon, J. Kim, H. Lee, H.-W. Suh, and D. Mun, "Construction of design requirements knowledgebase from unstructured design guidelines using natural language processing," *Comput. Ind.*, vol. 159–160, p. 104100, Aug. 2024, doi: 10.1016/j.compind.2024.104100.
[20] A. Sallam, A. Chetan, S. Mehrdad, L. C. Briand, and M. Traynor, "Automated demarcation of requirements in textual specifications: a machine learning-based approach," *Empir. Softw. Eng.*, vol. 25, no. 6, pp. 5454–5497, Nov. 2020, doi: 10.1007/s10664-020-09864-1.
[21] Y. Muhammad, J. D. N. A, G. Imran, and M. A. Shah, "Extraction of non-functional requirement using semantic similarity distance," *Neural Comput. Appl.*, vol. 32, no. 11, pp. 7383–7397, June 2020, doi: 10.1007/s00521-019-04226-5.
[22] Q. A. Shreda and A. A. Hanani, "Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches," *IEEE Access*, pp. 1–1, 2021, doi: 10.1109/ACCESS.2021.3052921.
[23] P. Jafari, M. A. Hattab, E. Mohamed, and S. AbouRizk, "Automated Extraction and Time-Cost Prediction of Contractual Reporting Requirements in Construction Using Natural Language Processing and Simulation," *Appl. Sci.*, vol. 11, no. 13, p. 6188, July 2021, doi: 10.3390/app11136188.
[24] S. Bashir, M. Abbas, M. Saadatmand, E. Enoiu, M. Bohlin, and P. Lindberg, "Requirement or Not, That is the Question : A Case from the Railway Industry," presented at the 29th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ 2023. Barcelona, Spain. 17 April 2023 through 20 April 2023, Springer Science and Business Media Deutschland GmbH, 2023, pp. 105–121. Accessed: Oct. 24, 2024. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:ri:diva-64397
[25] M. A. Khan, M. S. Khan, I. Khan, S. Ahmad, and S. Huda, "Non Functional Requirements Identification and Classification Using Transfer Learning Model," *IEEE Access*, vol. 11, pp. 74997–75005, Jan. 2023, doi: 10.1109/ACCESS.2023.3295238.
[26] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with Requirements Smells," *J. Syst. Softw.*, vol. 123, pp. 190–213, Jan. 2017, doi: 10.1016/j.jss.2016.02.047.
[27] P. S. Kummler, L. Vernisse, and H. Fromm, "How Good are My Requirements?: A New Perspective on the Quality Measurement of Textual Requirements," in *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, Coimbra: IEEE, Sept. 2018, pp. 156–159. doi: 10.1109/QUATIC.2018.00031.

[28] E. Alemneh and F. Berhanu, “Software Requirement Smells and Detection Techniques: A Systematic Literature Review,” *Cybern. Inf. Technol.*, vol. 24, no. 4, pp. 78–107, Dec. 2024, doi: 10.2478/cait-2024-0037.
[29] S. F. Tjong and D. M. Berry, “The Design of SREE — A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned,” in *Requirements Engineering: Foundation for Software Quality*, Springer, Berlin, Heidelberg, 2013, pp. 80–95. doi: 10.1007/978-3-642-37422-7_6.
[30] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno, “A framework to measure and improve the quality of textual requirements,” *Requir. Eng.*, vol. 18, no. 1, Art. no. 1, Mar. 2013, doi: 10.1007/s00766-011-0134-z.
[31] P. Saint Dizier and J. Kang, “LELIE - An Intelligent Assistant for Improving Requirement Authoring,” *Int. J. Requir. Eng.*, vol. 2015–02, pp. 1–9, 2015.
[32] C. Arora, M. Sabetzadeh, A. Goknil, L. C. Briand, and F. Zimmer, “NARCIA: an automated tool for change impact analysis in natural language requirements,” in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, Bergamo Italy: ACM, Aug. 2015, pp. 962–965. doi: 10.1145/2786805.2803185.
[33] M. Landhaußer, S. J. Korner, W. F. Tichy, J. Keim, and J. Krisch, “DeNom: a tool to find problematic nominalizations using NLP,” in *2015 IEEE Second International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, Aug. 2015, pp. 1–8. doi: 10.1109/AIRE.2015.7337623.
[34] H. Yang, A. Willis, A. De Roeck, and B. Nuseibeh, “Automatic detection of nocuous coordination ambiguities in natural language requirements,” in *Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering*, in ASE ’10. New York, NY, USA: Association for Computing Machinery, Sept. 2010, pp. 53–62. doi: 10.1145/1858996.1859007.
[35] S. Panichella and M. Ruiz, “Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback,” in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, Aug. 2020, pp. 404–407. doi: 10.1109/RE48521.2020.00057.
[36] R. Pinquié, P. Véron, F. Segonds, and N. Croué, “A requirement mining framework to support complex sub-systems suppliers,” *Procedia CIRP*, vol. 70, pp. 410–415, Jan. 2018, doi: 10.1016/j.procir.2018.03.228.
[37] M. Z. Qureshi, A. Azhar, Q. Abubakar, T. A. Rana, and A. Maqbool, “Capturing Users Requirements Using a Data Mining Approach,” in *2021 International Conference on Communication Technologies (ComTech)*, Sept. 2021, pp. 49–54. doi: 10.1109/ComTech52583.2021.9616939.
[38] Z. Wang, Chen ,Chun-Hsien, Zheng ,Pai, Li ,Xinyu, and L. P. and Khoo, “A graph-based context-aware requirement elicitation approach in smart product-service systems,” *Int. J. Prod. Res.*, vol. 59, no. 2, pp. 635–651, Jan. 2021, doi: 10.1080/00207543.2019.1702227.
[39] M. Shibaoka, H. Kaiya, and M. Saeki, “GOORE : Goal-Oriented and Ontology Driven Requirements Elicitation Method,” in *Advances in Conceptual Modeling – Foundations and Applications*, Springer, Berlin, Heidelberg, 2007, pp. 225–234. doi: 10.1007/978-3-540-76292-8_28.
[40] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, I. Omoronyia, and H. Zojer, “Ontology-Driven Guidance for Requirements Elicitation,” in *The Semanic Web: Research and Applications*, Springer, Berlin, Heidelberg, 2011, pp. 212–226. doi: 10.1007/978-3-642-21064-8_15.
[41] S. Reddivari, T. Bhowmik, and C. Hollis, “Automated support to capture verbal just-in-time requirements via audio mining and cluster-based visualization,” *J. Ind. Inf. Integr.*, vol. 14, pp. 41–49, June 2019, doi: 10.1016/j.jii.2018.06.001.

[42] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, “A recommender system for requirements elicitation in large-scale software projects,” in *Proceedings of the 2009 ACM symposium on Applied Computing*, in SAC ’09. New York, NY, USA: Association for Computing Machinery, Mar. 2009, pp. 1419–1426. doi: 10.1145/1529282.1529601.

[43] N. Yusop, M. Kamalrudin, M. M. Yusof, and S. Sidek, “Meeting Real Challenges in Eliciting Security Attributes for Mobile Application Development,” *J. Internet Comput. Serv.*, vol. 17, no. 5, pp. 25–32, 2016, doi: 10.7472/jksii.2016.17.5.25.

[44] M. Kamalrudin, J. Grundy, and J. Hosking, “Managing Consistency between Textual Requirements, Abstract Interactions and Essential Use Cases,” in *2010 IEEE 34th Annual Computer Software and Applications Conference*, July 2010, pp. 327–336. doi: 10.1109/COMPSAC.2010.40.

[45] J. Bhatia, R. Sharma, K. K. Biswas, and S. Ghaisas, “Using Grammatical Knowledge Patterns for structuring requirements specifications,” in *2013 3rd International Workshop on Requirements Patterns (RePa)*, July 2013, pp. 31–34. doi: 10.1109/RePa.2013.6602669.

[46] A. Rago, C. Marcos, and J. A. Diaz-Pace, “Uncovering quality-attribute concerns in use case specifications via early aspect mining,” *Requir. Eng.*, vol. 18, no. 1, pp. 67–84, Mar. 2013, doi: 10.1007/s00766-011-0142-z.

[47] V. Antinyan and M. Staron, “Rendex: A method for automated reviews of textual requirements,” *J. Syst. Softw.*, vol. 131, pp. 63–77, Sept. 2017, doi: 10.1016/j.jss.2017.05.079.

[48] V. Ambriola and V. Gervasi, “On the Systematic Analysis of Natural Language Requirements with CIRCE,” *Autom. Softw. Eng.*, vol. 13, no. 1, pp. 107–167, Jan. 2006, doi: 10.1007/s10515-006-5468-2.

[49] A. Ferrari and A. Esuli, “An NLP approach for cross-domain ambiguity detection in requirements engineering,” *Autom. Softw. Engg*, vol. 26, no. 3, pp. 559–598, Sept. 2019, doi: 10.1007/s10515-019-00261-7.

[50] A. Sardinha, R. Chitchyan, N. Weston, P. Greenwood, and A. Rashid, “EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements,” *Autom. Softw. Eng.*, vol. 20, no. 1, pp. 111–135, 2013, doi: 10.1007/s10515-012-0106-7.

[51] T. H. Nguyen, B. Q. Vo, M. Lumpe, and J. Grundy, “REInDetector: a framework for knowledge-based requirements engineering,” *Proc. 27th IEEEACM Int. Conf. Autom. Softw. Eng.*, 2012, doi: 10.1145/2351676.2351754.

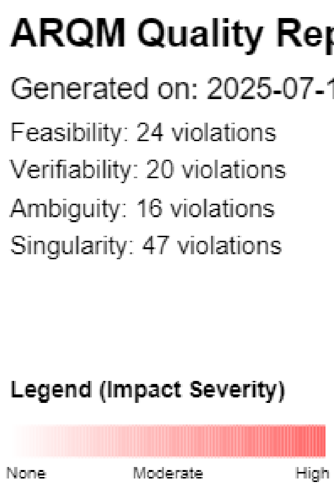
[52] W. Miao *et al.*, “Automated Requirements Validation for ATP Software via Specification Review and Testing,” in *Formal Methods and Software Engineering*, K. Ogata, M. Lawford, and S. Liu, Eds., Cham: Springer International Publishing, 2016, pp. 26–40. doi: 10.1007/978-3-319-47846-3_3.

[53] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, “Improving agile requirements: the Quality User Story framework and tool,” *Requir. Eng.*, vol. 21, no. 3, pp. 383–403, Sept. 2016, doi: 10.1007/s00766-016-0250-x.

[54] E. Parra, C. Dimou, J. Llorens, V. Moreno, and A. Fraga, “A methodology for the classification of quality of requirements using machine learning techniques,” *Inf. Softw. Technol.*, vol. 67, pp. 180–195, Nov. 2015, doi: 10.1016/j.infsof.2015.07.006.

Appendix

ARQM Visualization



ARQM Quality Report

Generated on: 2025-07-16 16:27:23

Feasibility: 24 violations

Verifiability: 20 violations

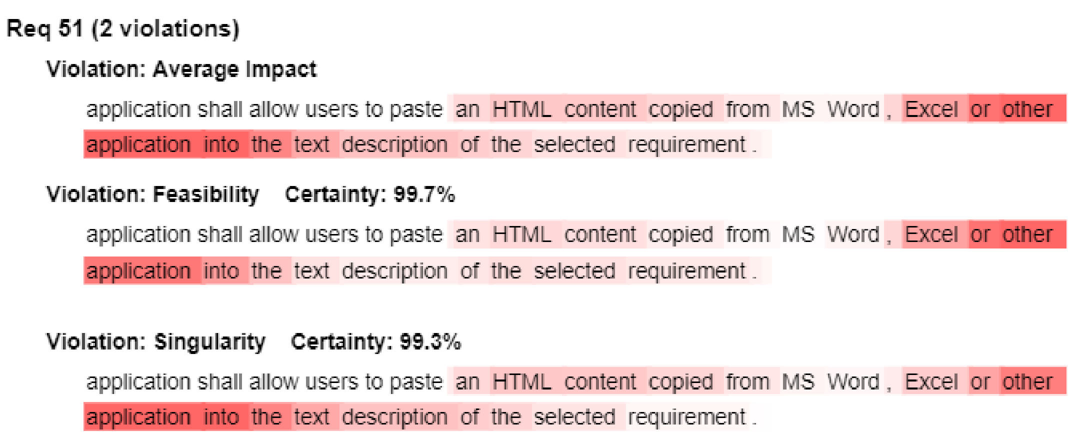
Ambiguity: 16 violations

Singularity: 47 violations

Legend (Impact Severity)

None Moderate High

Fig. 7 *ARQM Aggregated Quality Results*



Req 51 (2 violations)

Violation: Average Impact

application shall allow users to paste an HTML content copied from MS Word , Excel or other application into the text description of the selected requirement .

Violation: Feasibility Certainty: 99.7%

application shall allow users to paste an HTML content copied from MS Word , Excel or other application into the text description of the selected requirement .

Violation: Singularity Certainty: 99.3%

application shall allow users to paste an HTML content copied from MS Word , Excel or other application into the text description of the selected requirement .

Fig. 8 *ARQM SHAP Individual Requirement Result*