Jestr

Journal of Engineering Science and Technology Review 15 (2) (2022) 215 - 227

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Identification of Data-Intensive Systems Requirements using Semantic Similarity Search

Renita Raymond and S. Margret Anouncia*

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

Received 4 February 2022; Accepted 6 May 2022

Abstract

The phenomenal growth of big data in social applications and IT software platforms over the last few decades has emphasized the significance of a systematic requirement engineering strategy for analyzing the requirements of data-intensive systems, that deliver valuable insights to business entities. Classification of data-intensive requirements can aid in the development of a more systematic and transparent requirements engineering process, resulting in increased requirement compliance and software project completion. As a result, this paper provides a unique approach Word2Vector based Fast Similarity Search (WV-FaSS) for improving the process of software requirement categorization for data-intensive systems in two phases. Word2Vec begins by taking a corpus of software requirements as input and producing a well-trained high-dimensional vector space. Following that, the vectors that were extracted semantically are indexed. The query vector is then used to look for the most similar vectors within the index, and lastly, similar documents are obtained. Experiments on two benchmark datasets, PURE and WARC, as well as a dataset from the private IT industry, demonstrated that our model outperformed state-of-the-art techniques with precision, recall, and F1 values of 0.91, 0.9, and 0.9, respectively. Thus, the proposed model WV-FAISS enables developers to rapidly search for embeddings of similar requirements that are similar to one another, while also increasing the scalability of similarity search methods.

Keywords: Data-intensive requirements, Word2Vector, FaSS, Requirement Classification, Sematic Similarity Search

1. Introduction

In essence, big data is a vast collection of unstructured and organized data that is challenging to process using conventional methods [1]. By evaluating this vast amount of data, Data-Intensive Applications (DIA) assist corporate organizations in making predictive and informed recommendations. Business value is discovered through the requirements for big data applications. Thus, to elicit software requirements for the DIA, it is crucial to define the projects' implications early on [2]. Developing data-intensive systems necessitates the collection of specialized requirements for massive data. As a result, Requirement Engineering (RE) enables collaboration with diverse stakeholders and business analyst experience in analytical thinking to identify and adhere to the value and importance of each requirement. Because, according to statistics, RE is responsible for 60% of software development errors. As a result, gathering pertinent requirements reduces the risk associated with software-intensive initiatives and hence increases quality [3]. Furthermore, requirements are iterative, dynamic, interactive, and never-ending [4]. Since the majority of requirements are expressed in natural language, developers, analysts, and software architects always strive to manually classify them, as it is time-consuming and error-prone. These activities necessitate specialization, education, experience, and domain knowledge [5]. Developers can organize and structure requirements for feature extraction, classification, and speech recognition by applying Natural Language Processing (NLP). Appropriate requirement classification based on the Software Requirement Specification (SRS) improves the quality of software-intensive products [6]. Nonetheless, the requirements engineering approach for traditional and big data business intelligence systems is similar in many ways and distinct in others. To comprehend and classify the requirements for end-user applications, a very clear description is required [7].

In DIA, requirements must be processed independently and precisely categorized to increase requirement quality and minimize budget overruns. It is necessary to develop techniques for automatically classifying the elicited requirements into distinct classes [8]. According to Manal et al. [9], machine learning (ML) approaches to requirement document classification outperform classical natural language processing approaches. However, a systematic level of comprehension remains insufficient. Consequently, diverse classification schemes are utilized to differentiate between functional and non-functional needs [10]. However, there was no automated technology to enable the analysis and management of data-intensive requirements, which resulted in a variety of negative implications for DIA, including budget overruns, quality and security concerns, and customer unhappiness. Additionally, since the amount of data generated on the internet continues to grow exponentially, it is difficult for developers to identify and extract relevant information from the SRS, particularly textual requirements, due to their complex semantic meaning. Nonetheless, building DIA is a more difficult process as the corpus to be classified grows to millions of petabytes daily on the internet. As a result, a novel technique called WV-FAISS is proposed for classifying and retrieving high-dimensional vectors. Word2Vec converts an unstructured source corpus to labeled data and then learns how words are represented in a classification challenge. Data may be supplied into the model

*E-mail address: smargertanouncia@vit.ac.in
ISSN: 1791-2377
doi:10.25103/jestr.152.25

in real-time and requires minimal filtering, requiring little memory. and also maintains the semantic links between the vectors. Even novices can grasp the concept and carry it out. As a result, a fine-grained technique called WV-FaSS takes use of data parallelism by parallelizing the processing of individual items, or words in a single document. This significantly enhances the search for k nearest neighbors in textual datasets and is easily mappable to newer highly threaded accelerators such as manycore GPUs.

The Business Analysis Book of Knowledge (BABOK) classifies requirements into four categories [11], as illustrated in Fig. 1.

- **Business Requirements** – assertions of goals, objectives, and outcomes that justify initiating a change.
- **Stakeholder Requirements** – enumerate the stakeholders' requirements that must be addressed to fulfill the business requirements.
- **Solution Requirements** – define the features and characteristics of a solution that satisfies stakeholder requirements.
- **Transition Requirements** - specify the competencies and circumstances that the solution must meet to facilitate the transition from the present to the future state, but which are no longer required after the change is accomplished.

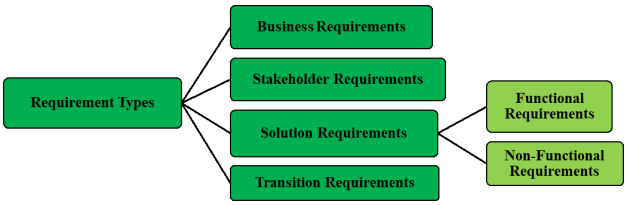

Fig. 1. BABOK Classification Schema

The research focuses on the distinction between Business Requirements (BR), Stakeholder Requirements (STR), Solution Requirements (SR), and Transition Requirements (TR), a subject that has been extensively investigated and whose theoretical foundations are currently being debated. The study mainly aims to retrieve the transition requirements for real-life scenarios. The rationale for emphasizing the importance of retrieving transition requirements is that neglecting their presence can have a significant real-world impact, resulting in a deteriorating product. They differ from other sorts of requirements in that they cannot be specified until a solution has been constructed. These requirements have a shorter lifespan than others since they only apply during solution transitions. Transition needs are handled in the same way as other requirements. The changes are in the sources, the type of changeover demands, and the fact that they become obsolete after the preceding solution is eliminated.

The following contributions are presented in the paper.

- We manually annotate 3500+ requirements from the widely used PURE dataset [12], the WARC dataset [13], and one industrial project. Our annotations are based on the BABOK taxonomy [11], which allows a requirement to include organizational needs as well as functional and quality characteristics.
- To define the baseline, annotated requirements are pre-processed and extracted into high dimensional feature vectors using Word2Vec. The vectors that were extracted semantically are indexed using FaSS. The query vector is then used to look for the most similar vectors within the index, and lastly, similar documents (transition requirements) are obtained.
- The proposed WV-FAISS framework is validated on the benchmark datasets PURE and WARC and the performance measures are shown.

2. Related Work

This section describes related work on software requirement classifications and provides an outline of how semantic similarity search is used in the field of big data software engineering.

According to studies, a failure to comprehend and classify requirements is the primary cause of budget and time overruns, resulting in software system failure. All software requirements must be expressed completely and consistently, including specifications of all necessary services. On the other hand, developers must carefully read, interpret, and discern those requirements [14]. Initially, the process of writing and classifying software was carried out manually. Numerous modules and methodologies have been used to automate the process of requirement writing and classification over the last two centuries. Information Retrieval (IR), Genetic Algorithms, and Clustering Algorithms, among others, have been used to solve software requirement problems [15, 24]. All of the previous state-of-the-art methodologies are contingent on the developer's experience and background. This creates difficulties for the business and its stakeholders as a result of potential complications.

The success of a software system is highly dependent on compliance with non-functional requirements because when they are overlooked or neglected, severe complications develop. Slank et al. [25] offered a tool-based strategy to address this issue, namely the NFR finder. This tool categorizes and extracts sentences from natural language documents according to their NFR category. While the NFR finder enables analysts to effectively extract NFRs from available natural language materials via automated natural language processing, it is limited to text. It is unable to parse images and tables contained within the unconstrained document currently open. Similarly, security-related issues must be carefully examined while developing software that fulfills the needs of the consumer. Security requirements have been classified using text mining techniques and prediction models [26]. In 2017, Liang et al. [27,28] used feature extraction and machine learning techniques to automatically categorize user review needs, concluding that AUR -BoW with Bagging achieves the best classification results. Using semi-supervised and unsupervised machine learning methods, requirements can also be accurately identified as FRs and NFRs.

Additionally, a semi-supervised classification algorithm can be utilized to automatically extract the FR and NFRs from the SRS. In comparison to supervised techniques, semi-supervised techniques produce superior outcomes because they use only a little amount of labeled data. In the former, all data sets must be labeled to facilitate classification. One such example is the app store, where requirements are identified as functional or non-functional using a self-labeling algorithm as

part of a semi-supervised classification technique [29]. Semi-supervised classification approaches to aid in appropriately classifying requirements. Additionally, it will be enhanced in the future through the use of unsupervised learning approaches.

2.1 Requirement Pre-processing

SRS is composed of massive amounts of data of various types, all of which are heterogeneous by nature and include inconsistent values. Pre-processing is a critical step that must be accomplished prior to utilizing the data for model training. Tokenization, stop words removal, error correction, normalization, and vectorization are the primary pre-processing stages [30,32]. Uysal et al. [31] tested the effectiveness of a mixture of pre-processing approaches on two domains, e-mail, and news, in two distinct languages. The results indicate that depending on the domain and language investigated, selecting optimal combinations of pre-processing activities greatly enhances classification accuracy. It is self-evident that pre-processing results in cleaner, more manageable data sets, which are required for any business organization to gain significant insights.

2.2 Feature Extraction

The feature extraction procedure converts text into feature vectors using NLP pre-processing techniques. It enhances the learning algorithm's accuracy and speed. Therefore, this section discusses the various feature extraction strategies and their limitations. Using the vector space approach to choose features decreases to feature space dimensions [33]. With a predetermined keyword set, feature extraction techniques such as TF-IDF, Bag of Words, and Word2Vec generate the weights of the words in the text [34]. One hot encoding technique transforms text into a vector by building a vocabulary. However, due to its memory requirements, it cannot collect contextual information [35].

The BoW is a simple and effective feature extraction technique. In BoW, texts are represented as a bag of words by counting the number of occurrences of each instance or word in the bag, regardless of sequence or grammar. Bow assigns a value to each feature in the document, giving them equal weight. The model is also influenced by recurring elements rather than the document's relevance. But its non-zero dimensions and big vocabulary size lead to a high sparse and dimensional feature vector [36,37]. TFIDF is determined by multiplying the term and inverse document frequencies, according to Qaisier et al. [38]. Terms having a high TF-IDF weight are deemed to be more significant than those with a lower TF-IDF score. Though TF-IDF is the most well-known and widely used formula for generating a vector descriptor with several normalized forms, it has certain disadvantages. TFIDF is unconcerned about a term's position in the text, its meaning, or its co-occurrences with other texts in the document. To address the limitations of TF-IDF, an extended form of Fuzzy based TF-IDF (FTF-IDF) is introduced in 2019. FTF-IDF is a vector representation in which the TF-IDF components are supplied to the Fuzzy Inference System as inputs (FIS). Following the defuzzification stage, weight terms are created as crisp outputs. FTF-IDF assigns semantic meanings to the documents' terms [39]. However, it does not attempt to investigate the co-occurrences of other texts inside the documents.

In the same year, Lakshmi et al. [40] introduced term weighting approaches to represent text documents using Term Frequency - Ranking of Term Frequency (TF-RTF) and Term Frequency - Ranking of fuzzy logic with the semantic relationship of terms (TF-RFST). It outperforms word count, Term Frequency-Inverse Document Frequency (TF-IDF), Term Frequency-Inverse Corpus Frequency (TF- ICF), Multi-Aspect TF (MATF), BM25 in terms of accuracy, recall, and F1 measure. However, it does not focus on the syntax of the phrases. Also, Ricardo et al. [41] started YAKE without a trained huge corpus. It supports many languages and documents of any length. But it can't find explicitly assigned keywords in the text. Okapi BM25 is a ranking function that estimates document relevance to a search query regardless of document proximity. Using Twitter data, Kadhim et al. [42] found that TF-IDF outperforms BM25 in terms of F1 measure. A large corpus cannot use BM25.

Since the weighting process is simply a linear transformation of feature vectors, adopting a weighting strategy is not required. In a nutshell, researchers can utilize any of the text feature extraction strategies or a hybrid of techniques based on the requirements of their study, as each method has its pros and cons [43].

2. 3 Software Requirement Classification

The sub-section consists of various classification techniques suggested by the researchers to classify the requirements automatically. In 2019, Rahman et al. [44] used a variety of machine learning approaches to completely extract NFR from the SRS document. According to the statistical study, the SVM classifier produces the best results with a precision of 0.66, a recall of 0.61, and an accuracy of 0.76. The trials used the well-known PROMISE dataset, which exhibits an imbalanced distribution of FRs and NFRs. Lima et al. [45] developed the PROMISE exp repository by expanding the PROMISE dataset. Once more, Edna et al. [46] conducted a comparative analysis of various machine learning algorithms such as Support Vector Machine (SVM), KNN (K Nearest Neighbour), Decision Tree, Multinomial Naive Bayes (MNB), and Logistic Regression (LR) to determine which algorithm is the best fit for automatically classifying requirements using PROMISE exp. The results indicate that the combination of TF-IDF with LR produces the best performance measures, with an F-measure of 91% for binary classification, 74% for 11-granularity classification, and 78% for 12-granularity classification. Before conducting any experimental analysis, researchers must verify whether the dataset being used is balanced or unbalanced because an unbalanced dataset leads to poor automatic classification of requirements.

Fuzzy Rough Set (FRS) is a sophisticated mathematical technique for handling uncertain data. The Fuzzy Rough Set based on Robust Nearest Neighbor (FRS-RNN) was proposed by Behera et al. [47]. Documents are first extracted using a modified CNN, then categorized using FRS-RNN. It outperforms SVM, Naive Bayes, DNN, and CNN. However, FRS-RNN hyperparameter tuning takes longer than other machine learning methods. An NFR sentence can have multiple classes. In 2019, Fuzzy Similarity KNN (FSKNN) was proposed for multi-label requirement categorization based on ISO/lEe 25010. On the other hand, the fuzzy similarity measure technique is employed to obtain a training pattern. The training data search set is utilized to find the K nearest neighbor. A maximum a posteriori (MAP) estimate will be used to categorize a test document [49]. A semi-supervised classification strategy was employed to classify the FR and NFR contained in the reviews on the APP store. The self-labeling algorithm assigns labels to the unlabelled data and classifies future reviews that aren't yet

seen. The findings, however, have not been empirically tested [49].

In the field of RE, semantic information is extremely important. To create higher-quality semantic-based SRS, software engineers apply effective requirement classification approaches. For sharing and characterizing the classifications of requirements, a Requirement Classification Ontology (RCO) has been created. It is used to verify the semantic correctness of the RE process, maintaining consistency between the requirements [50]. In comparison to fuzzy rule mechanisms, machine learning approaches play a significant role in classifying requirements like FRs, NFRs, quality requirements, security requirements, legal requirements, and so on, according to various studies [51,52,53]. In 2020, Manuel et al. [54] found that Naive Bayes, K Nearest Neighbor, J48, and Natural Language Processing methods are the most frequently used classification algorithms. Academic databases and user reviews are the most popular training datasets.

Categorizing requirements by business organizational needs will enhance transparency in the RE process, promoting requirement fulfillment and finishing software-intensive projects. Contrary to popular belief, no research has been done to address the issues of extracting data-intensive system needs or to define criteria for categorizing requirements based on interactions. So, considering the technology's utility in software requirement classification, a new framework based on the BABOK classification schema [11] is established. Limiting the criteria to transition type, in particular, allows the engineers to focus on what the DIA developers care about while focusing on the ways to achieve it. To our knowledge, none of the existing techniques substantiate the resulting requirement categories. This section's literature review emphasized the significance of semantic analysis search for identifying and retrieving data-intensive system requirements. Additionally, it demonstrated the inability of existing methodologies to produce satisfactory results with diverse datasets. Therefore, the proposed approach introduces a novel framework WV-FAISS for categorizing requirements and validating the categorized requirements.

3 Classification Problem – Revisited

Based on the BABOK classification schema [11], the classification problem is revisited as shown in Fig.2 and is applied to PURE [12], WARC [13] datasets, and one banking dataset obtained from the private sector. The PURE dataset consists of 79 SRS with 34,268 sentences. WARC dataset, web archive tool comprising of 89 SRS with at least 21,456 sentences. The private dataset consists of 1512 requirements. We have selected 6876 requirements in total from the dataset and classified them as BR, STR, SR, and TRs.

Each dataset was individually labeled by two authors. The taggers then convened reconciliation meetings to resolve tagging disagreements. If the taggers were unable to agree on a final tag, a third author was consulted. The authors resolved all disagreements. Krippendorff's alpha (α) metric is a reliability coefficient developed to quantify the agreement between observers, coders, judges, or measuring equipment when making distinctions or assigning numerical values to normally unstructured events [55]. Table I shows the value of Krippendorff's alpha (α) for the tagged dataset. On average, $\alpha \geq 0.8$ means the perfect agreement of the dataset by the raters.

The output of the tagging procedure following the reconciliation sessions are summarised in Table II and Fig.2. The datasets are sorted by the number of rows in each requirement. As previously stated, the taggers assigned the tags BR, STR, SR, and TR. Using these four identifiers, we next determined whether the row contains only BR, STR, SR, or TR. Following that, the reconciled classification is used to train and test the classifiers.

Table 1. Krippendorff's alpha (α) for the datasets

Data set	Number of Sentences	Business Requirements (BR)	Stakeholder Requirements (STR)	Solution Requirements (SR)	Transition Requirements (TR)
PURE (Data set 1)	4352	0.79	0.81	0.86	0.81
WARC (Data set 2)	695	0.84	0.83	0.78	0.89
Banking (Data set 3)	1829	0.80	8.87	0.81	0.84

Table 2. Overall Tagged Dataset

Data set	Number of Sentences	Business Requirements (BR)	Stakeholder Requirements (STR)	Solution Requirements (SR)	Transition Requirements (TR)
PURE (Data set 1)	4,352	452	1,089	1,826	986
WARC (Data set 2)	695	97	187	278	133
Banking (Data set 3)	1,829	251	479	582	517
Total	6,876	800	1,755	2,686	1,636

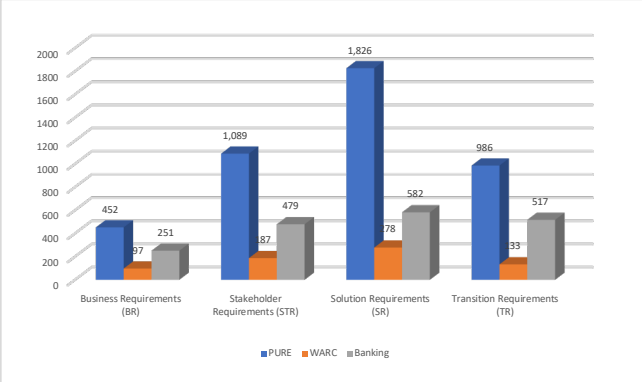

Fig. 2 Distribution of Requirements over the datasets

Generally, there is some degree of imbalance between classes in any real data set. If the level of imbalance is pretty modest, there should be no impact on the performance of the knn classifier. As illustrated in Fig. 2, there is a considerable degree of imbalance between requirement categories in our dataset (classes). This is a frequent occurrence in requirements categorization. This issue prompted us to employ a variety of strategies for balancing the dataset to produce trustworthy classification results. The re-sample

technique is one of the most often used techniques for text instance balance. This strategy is based on over-sampling minority groups and under-sampling majority groups. To generate its synthetic data, this technique used the SMOTE strategy, which is based on the concept of nearest neighbors. SMOTE provides synthetic samples and instances for minority classes. This inevitably results in the creation of additional data that is identical to what we already have, without adding any diversity to our dataset. While under-sampling procedures are achieved by randomly deleting a percentage of instances. Following the balancing task, the dataset became adequately balanced across categories, as illustrated in Fig. 3.

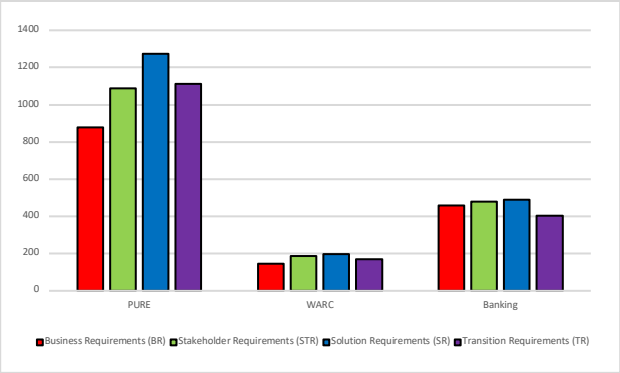

Fig. 3 Distribution of Balanced Datasets to categories

4 Implementation

The Proposed design is shown in Fig. 4 which describes the four-phase implementation process. It starts from gathering the requirements from the stakeholders, business analysts, and later on, with the final approval of requirement analyst SRS is documented. In the second phase, SRS is given as an input for document pre-processing, then the features are extracted into high dimensional vectors using semantic embedding Word2Vec. Following that, the vectors that were extracted semantically are indexed using FaSS. The query vector is then used to look for the most similar vectors within the index, and lastly, similar documents (requirements) of various classes are obtained.

4.1 Requirement Elicitation

Requirement elicitation is used during the RE phase to elicit requirements for creating software-intensive projects from users, consumers, and other stakeholders. The requirements for DIAs should be identified early in the software life cycle. Conventional RE methods are incapable of meeting the organization's needs for two primary reasons. To begin, it is primarily concerned with generic user requirements and provides no real insight into the features provided by big data that contribute to a more effective business intelligence solution. Second, the massive volume of data generated daily by various systems increases demand for various types of consumption. As a result, business analysts are included in the requirement elicitation process for DIAs to give business intelligence solutions to organizations.

During the framework's early phase, a form was created to collect and document requirements from diverse stakeholders. The stakeholder form used to collect requirements is depicted in Figure 5. The form captures various details about a requirement, such as a stakeholder's name, their role, the purpose of the requirement, the data required for the requirement, the stakeholder's status, whether primary or secondary, the mode of interaction when entering the requirement, and the requirement's description. As mentioned earlier, one dataset (banking) obtained from the private sector is gathered using the stakeholder form.

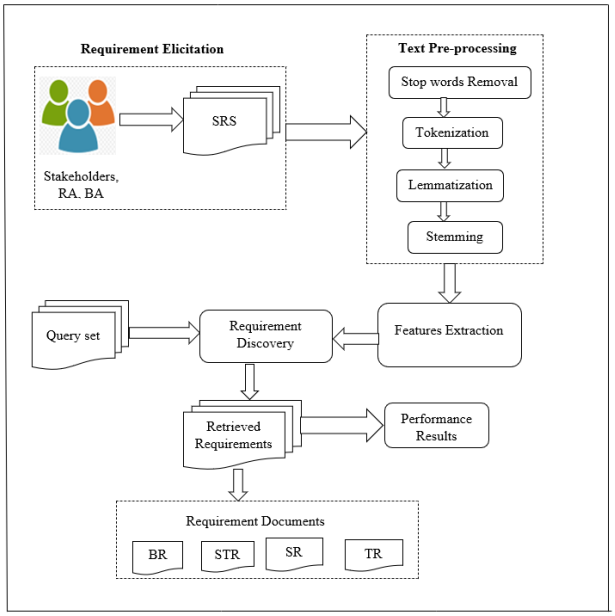

Fig. 4 Proposed Framework of Requirement Classification using WV-FAISS

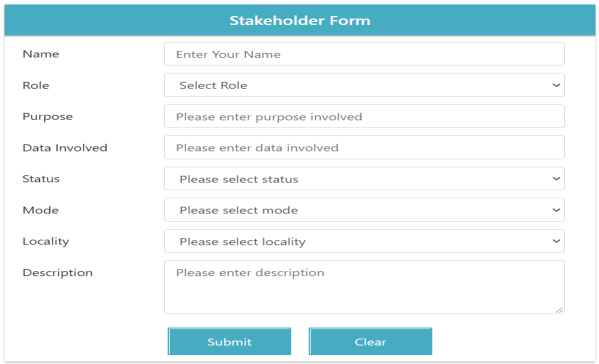

Fig. 5. Stakeholder Form

Requirement analyst determines the gap between what customers require and what the project stakeholders require, validates, and documents those requirements. During the analysis phase, the analyst classifies the requirements received via stakeholder form as stable or volatile in terms of their priority and feasibility [56]. The sample of requirements collected for the banking dataset is approved by the requirement analyst and is shown in Fig.6.

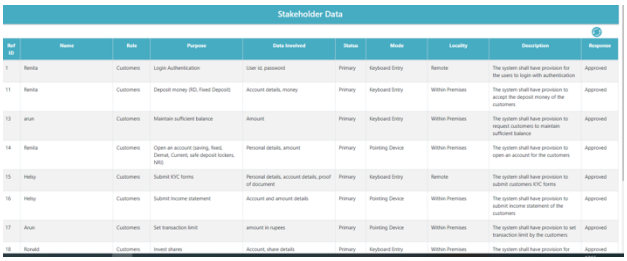

Fig. 6. Sample of Requirements Approved by Requirement Analyst

4.2 Requirement Pre-Processing

Requirement Pre-Processing is the second stage of the classification process. It immediately enhances model performance by reducing noise or ambiguous data taken from various sources. A series of actions are done to standardize textual data so that it can be used as an input by analytics systems and apps. Numerous pre-processing techniques such as stop word removal, tokenization, stemming, and lemmatization is available to help categorize requirement documents. The SRS text has been tokenized to make it more meaningful. Predefined stop words are eliminated after the conversion to meaningful tokens. Occasionally, even stop words can be defined by the user according to their intended usage. Eliminating such terms from the corpus decreases the dimension of the term space, which improves the model's performance. Stemming is then used to determine the origin of a token within the corpus. This technique eliminates numerous suffixes, thus lowering the corpus tokens to save time and memory. Finally, lemmatization takes into account the morphological analysis of the tokens or words, reducing noise and accelerating the user's task [57, 58].

Table 3. Corpus Pre-Processing

RID	Requirement Description	Tokens
1	The system shall have provision for the users to login with authentication	['user', 'login', 'authentication']
2	The system shall have provision to accept the deposit money of the customers	['accept', 'deposit', 'money', 'customer']
3	The system shall have provision to request customers to maintain sufficient balance	['request', 'customer', 'maintain', 'sufficient', 'balance']
4	The system shall have provision to open an account for the customers	['open', 'account', 'customer']
5	The system shall have provision to submit customers KYC forms	['submit', 'customer', 'kyc', 'form']
6	The system shall have provision to submit income statements of the customers	['submit', 'income', 'statement', 'customer']
7	The system shall have provision to set transaction limits for the transactions by the customers	['set', 'transaction', 'limit', 'transaction', 'customer']
8	The system shall have provision for the customers to invest shares	['customer', 'invest', 'share']
9	The system shall have provision for the users to pay automated bill payments	['user', 'pay', 'automated', 'bill', 'payment']
10	The system shall have provision for the users to pay taxes	['user', 'pay', 'tax']
11	The system shall have provision for the users to recharge the data card	['user', 'recharge', 'data', 'card']
12	The system shall have provision for the customers to pay for travel through UPI	['customer', 'pay', 'travel', 'upi']
13	The system shall have provision for the users to pay due (loan)	['user', 'pay', 'due', 'loan']
14	The system shall have provision for the users to pay service charges	['user', 'pay', 'service', 'charge']
15	The system shall have provision for the users to set the ATM, Mobile Pin, Net Banking transaction pin	['user', 'set', 'atm', 'mobile', 'pin', 'net', 'banking', 'transaction', 'pin']
16	The system shall have provision for the customers to calculate EMI for loan	['customer', 'calculate', 'emi', 'loan']
17	The system shall have provision for the customers to check the account balance of their account	['customer', 'check', 'account', 'balance', 'account']
18	The system shall have provision for the users to withdraw the amount from their account	['user', 'withdraw', 'amount', 'account']
19	The system shall have provision for the customers to view their weekly, monthly transaction details	['customer', 'view', 'weekly', 'monthly', 'transaction', 'detail']
20	The system shall have provision for the customers to submit their details	['customer', 'submit', 'personal', 'detail']

All requirements in the corpus have gone through a pre-processing step. Table 3 shows the requirements in the corpus before the pre-processing and after pre-processing. In this paper, Spacy, a free, open-source library for NLP is being used to process and understand a large volume of text. It performs the pre-processing steps and provides the fastest and more accurate syntactic analysis of any NLP released to date [59]. For example, the requirement specified in the second row of table 3 after pre-processing is converted into tokens.

4.3 Feature Extraction

The corpus after wrangling, cleaning up, and standardizing the textual requirements into a form (i.e., tokens) is taken up as an input for the feature extraction process. This stage converts the pre-processed corpus into machine-learnable numerical features representing the information contained in the requirements. Since the actual text is extremely complex and unstructured, each unique word or token is viewed as a distinct dimension, making classification algorithms difficult to apply. Word2Vec is capable of identifying correlations between words, both syntactic and semantic. The embedding vector is compact and versatile, and because it is unsupervised, it requires less human work to tag the data [60]. It accepts as input a huge corpus of tokens generated from the second phase of the normalization procedure shown in Table 3 and generates a vector space of unique tokens. Words in the

vector space that occur in familiar settings in the corpus are clustered together. The sample of semantic vectors obtained from the tokens is depicted in Fig. 7.

```
0      [0.06095517, 0.025397392, 0.0055965967, -0.006...
1      [-0.075332925, 0.0139850285, -0.025303327, -0....
2      [-0.04405474, 0.04869568, -0.039536633, -0.006...
3      [0.014313849, 0.039584193, -0.044255454, -0.01...
4      [-0.07391806, -0.04680717, -0.07138344, 0.0042...
                            ...
98     [-0.026260227, 0.041225273, 0.00037527832, -0....
99     [-0.077638224, 0.052212063, -0.01442979, 0.052...
100    [-0.124552995, 0.028047856, 0.022083702, -0.03...
101    [-0.11497229, 0.023809854, 0.025683139, -0.029...
102    [-0.089345165, 0.0655677, 0.042519, -0.0842831...
Name: desc_vec, Length: 103, dtype: object
```

Fig. 7. Sample of semantic embedded vectors generated using Word2Vec

Spacy [59] parses full blocks of text and assigns word vectors from the loaded models seamlessly in Fig. 6. Word2vec enhances the quality of features by taking into account the contextual semantics of words in a text, hence increasing the accuracy of machine learning and requirement categorization.

4.4 Requirements Discovery

Requirement discovery is the final phase of the implementation process. In it, the semantically extracted vectors are indexed in memory. After converting the query set to a vector, it is utilized to find the most similar vectors within the index, and finally, similar requirements are retrieved using Fast Similarity Search (FaSS). FaSS compares the similarity values of a query vector to term vectors and delivers the matrices that fulfill the query conditions. The datasets examined in this article are textual. As it contains textual data, an inverted index can be used to swiftly locate documents that are similar to the query. Owing to the inverted index, considerable space is conserved. To begin, an inverted index is a built-in memory, assuming that the dataset is static and fits in memory. Let ν denote the input dataset's vocabulary, that is, the collection of different terms contained within the source set of documents $Ɗ_{in}$. The input data set £ is a collection of distinct term-document pairs (t, d) that appear in the entire dataset, with t ∈ ν and d ∈ $Ɗ_{in}$. The inverted index is stored in an array of size |£|. Once the set £ is in memory, each pair is inspected in parallel, with the result that each time a word is visited, the number of documents in which it occurs (document frequency - df) is increased and put in the array df of size | ν |. On the df array, a parallel prefix-sum is performed by tracing each value to the sum of the phrases preceding it and saving the outcomes in the indexing array. Then, each value of the index array refers to the location of the matching initial element in the inverted index, which will be used to store all (t, d) pairs in the order specified by the term. Then, given a query q, the proximity between q and $Ɗ_{in}$ is determined using the cosine distance metric, and the top k documents that are closest to the query are retrieved. By selecting a subset of documents with a high degree of correlation to the query, a threshold can be established to eliminate any possibilities with a similarity value less than it. Thus WV-FaSS is a very fast and scalable tool for computing the top k nearest in high dimensional and sparse data.

Fig. 8. Word Cloud of the Query Set

Fig.8 illustrates the word cloud of the query set created to retrieve the similar requirements matched to it. The word cloud of the query set is based on the banking (Dataset 3) obtained from the private sector. A threshold value of greater than 0.75 is set to retrieve the most similar documents (requirements) resulting from the proposed WV-FaSS. Results obtained using dataset 3 are presented in Table 4, 5, 6, and 7 respectively. Table 4, 5, 6, and 7 shows the top retrieved documents of BABOK classification schema – BR, STR, SR, and TR with a similarity score. Results obtained using benchmark datasets are presented in section 5. As the proposed methodology mainly focuses on the retrieval of transition requirements Table 4 produces the top 15 documents retrieved with a similarity score greater than 0.75. Tables 5, 6, and 7 illustrate the top 5 documents retrieved for each category with a similarity score greater than 0.75 respectively.

Table 4. Top 15 Documents Retrieved for Transition Requirement Type with Threshold value > 0.75

S. No	RID	Requirement Description	Requirement Type	Similarity Score
1.	53	The system shall have provision to accept deposits from customers	Transition	0.95782
2.	11	The system shall have provision to accept the deposit money of the customers	Transition	0.95216
3.	103	The system shall have provision to Invest capital	Transition	0.94384
4.	18	The system shall have provision for the customers to invest shares	Transition	0.92737
5.	20	The system shall have provision for the users to pay taxes	Transition	0.89454
6.	24	The system shall have provision for the users to pay service charges	Transition	0.89127
7.	11	The system shall have provision to accept the deposit money of the customers	Transition	0.84769

S. No	RID	Requirement Description	Requirement Type	Similarity Score
8.	19	The system shall have provision for the users to pay automated bill payments	Transition	0.84128
9.	22	The system shall have provision for the customers to pay for travel through UPI	Transition	0.83179
10.	53	The system shall have provision to accept deposits from customers	Transition	0.81059
11.	11	The system shall have provision to accept the deposit money of the customers	Transition	0.80456
12.	68	The system shall have provision for the staff to update the system software	Transition	0.79854
13.	104	The system shall have provision to Observe the stock exchange market	Transition	0.77826
14.	73	The system shall have provision for the staff to Exchange currency with other banks in case of cash shortage	Transition	0.76891
15.	70	The system shall have provision for the staff to Link account details with Aadhar	Transition	0.75697

Table 5. Top 5 Documents Retrieved for Solution Requirement Type with Threshold value > 0.75

S. No	RID	Requirement Description	Requirement Type	Similarity Score
1.	48	The system shall have provision for the users to login with authentication	Solution	0.98721
2.	137	The system shall have provision to view billing details	Solution	0.96745
3.	107	The system shall have provision to Check the financial statement of a customer	Solution	0.95781
4.	89	The system shall have provision to check for locker facility	Solution	0.89745
5.	54	The system shall have provision for the customers to submit their details	Solution	0.85478

Table 6. Top 5 Documents Retrieved for Stakeholder Requirement Type with Threshold value > 0.75

S. No	RID	Requirement Description	Requirement Type	Similarity Score
1.	39	The system shall have provision to get shares details in the stock market	Stakeholder	0.97215
2.	65	The system shall have provision to set the Regulation of foreign exchange	Stakeholder	0.94578
3.	92	The system shall have provision to set the Regulation of money	Stakeholder	0.92147
4.	104	The system shall have provision to Monitor Rotation of cash	Stakeholder	0.87458
5.	2	The system shall have provision to Limit the financial, legal, and reputational risks	Stakeholder	0.81726

Table 7. Top 5 Documents Retrieved for Solution Requirement Type with Threshold value > 0.75

S. No	RID	Requirement Description	Requirement Type	Similarity Score
1.	47	The system shall have provision to view the response time for the customer queries	Business	0.97215
2.	23	The system shall have provision to check the monthly transactions bills for customers feedback and responses	Business	0.94578
3.	73	The system shall have provision to monitor the cash flow per day, per week	Business	0.92147
4.	99	The system shall have provision to check whether the customer receives the banking product services in time	Business	0.87458
5.	81	The system shall have to evaluate the risk report documentation concerning loan approvals	Business	0.81726

Table 8 and Fig. 9 depict the distribution of sample documents retrieved for a few queries. For example, the query "Get" retrieves 15 documents and queries "Deposit", "update", "check" and "display" retrieve 25, 18, 30, and 8 documents respectively.

Table 8. Sample Documents Retrieved for the Query set

S. No	Query	No of Documents Retrieved
1.	Get	15
2.	Deposit	25
3.	Update	18
4.	Check	30
5.	Display	8

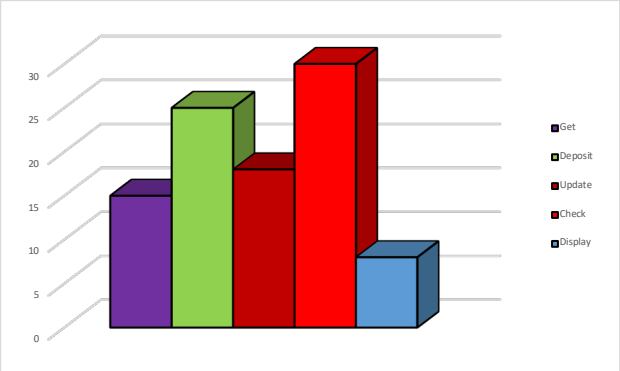

Fig. 9. Distribution of the Sample Documents Retrieved for Few Queries.

Proposed Algorithm WV-FaSS

Input: *let f represent the stakeholder form, SRS be the Software Requirement Specification, i be the i^{th} requirement in SRS*

Output: *let BR, STR, SR, and TR represent the Business Requirement, Stakeholder Requirement, Solution Requirement, Transition Requirement, respectively.*

Data: *Data set (x)*

Begin

Generate a stakeholder form f

foreach *f in the sequence do*

Get the requirements r_i from s € S where S= {Primary Stakeholder, Secondary Stakeholder}

Requirement Analyst Form ← Save r_i

RID ← Assign r_i // RID stands for Requirement ID

if *r_i is feasible and approved*

add r_i to SRS

else

revert to stakeholders

endif

endfor

Function Preprocessing *(SRS, Feature Vectors)*

Parse all the input requirements r_i where i = 1,2,3…. n

foreach *requirement r_i do*

Tokenize ← r_i

Store the Tokens as array

Create a customized stopword list

foreach *T from r_i*

compare T and customized stopword list

if *T = customized stopword list*

remove T from r_i

else

store the Tokens

Remove suffixes from the tokens

S_i ← Store tokens

endfor

endfor

Function FeatureExtraction *(S_i, WV)*

Let S_i be the tokens in corpus

Read the model word2vec

Set the parameters size =300, window = 2, min_count = 20, negative = 20, alpha = 0.03

foreach *S_i in the corpus do*

Build the vocabulary table v with d be the dimension of word vectors

Train the model

Return word vectors wv for the vectors S_i in the corpus

endfor

Function Query Processing *(wv, QS, ExD)*

Let QS_i be the Query Set where i = 1, 2…n, RD represent the requirement documents from SRS, ExD represent the extracted requirement documents

Create Query_Set (QS)

Create query vector qv € QS

foreach *query vector qv do*

compute distances between qv and WV

sort the computed distances

select the k-nearest vectors concerning k small distances (SimS)

if *qv = S_i in Corpus*

Retrieve the documents (RD_i) with SimS

else

Return no match

Assign ExD ← RD_i

endfor

end

5 Experimental Results

The experimental investigations were conducted on an Anaconda platform utilizing the Python programming language [62]. A Microsoft Windows Server 2012 R2 communication system was used in conjunction with an Intel Xeon E5-2630 2.20 CPU and 64 GB of memory. A comparative evaluation of several approaches is performed in terms of precision, recall, and F1 measure on the benchmark datasets PURE and WARC. The results demonstrate that, across all benchmark datasets (PURE, WARC), the proposed approach WV- FaSS consistently outperforms the competition in terms of precision, recall, and F1 measure compared to TF-FaSS (Term Frequency- Fast Similarity Search) [61], Question Bank Similarity Searching System (QB3S) [63], Rider Spider Monkey Optimization Algorithm (RSOA) [64], Document Ranking Optimization (DROPT) [65], Fuzzy Logic IR (Information Retrieval) [66] respectively.

5.1 Evaluation Metrics

Evaluation metrics are mainly used to evaluate a model's performance. The performance of the proposed model WV-FaSS is validated using mathematical methods that compare the model's predictions to the database's actual values.

Precision is defined as the proportion of retrieved documents that are relevant to the user's information need. Precision can be formulated as in equation 1.

$$P = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retreived\ documents\}|} \quad (1)$$

A Recall is the proportion of documents that are successfully retrieved relevant to the query. Recall can be formulated as in equation 2.

$$R = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}}{|\{relevant\ documents\}|} \quad (2)$$

F – measure is the weighted harmonic mean of precision and recall. F1 score can be formulated as in equation 3.

$$F = 2 * \frac{(P*R)}{(P+R)} \quad (3)$$

Table 9. Precision, Recall, and F – measure Comparison

S. No	Algorithm	Precision	Recall	F - measure
1.	WV-FaSS	**0.91**	**0.89**	**0.9**
2.	TF-FaSS	0.89	0.86	0.87
3.	QB3S	0.78	0.76	0.77
4.	RSOA	0.82	0.81	0.81
5.	DROPT	0.84	0.83	0.83
6.	Fuzzy Logic IR	0.79	0.78	0.8

Table 9 shows the weighted average scores where the proposed WV-FaSS depicts the highest precision, recall, and F1 measures compared to other state of art approaches like TF-FaSS, QB3S, RSOA, DROPT, and Fuzzy Logic IR. Fig. 10, 11, and 12 illustrates the comparison of weighted average scores.

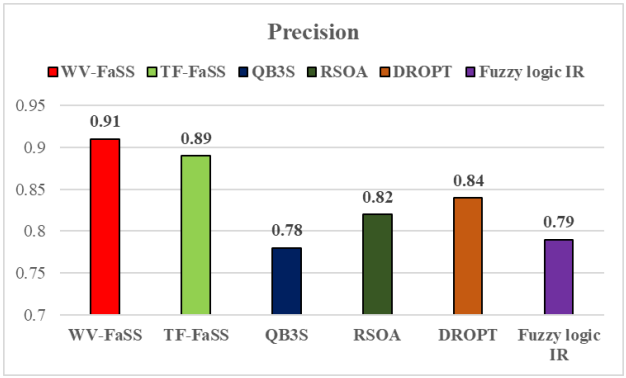

Fig. 10. Comparison of Precision Scores

5.2 Analysis of Computation Time

Computation time is measured to process the SRS requirements concerning the query set. It is the total time spent on the calculating process for the user input query. Seconds are used to denote the computing time (s) and are expressed in equation 4.

$$Computation\ time = \sum_{n=1}^{N}(computing\ time\ N\ queries) \quad (4)$$

Where,
N – Total number of queries
Computing time N queries – Time consumed for computing "N" queries

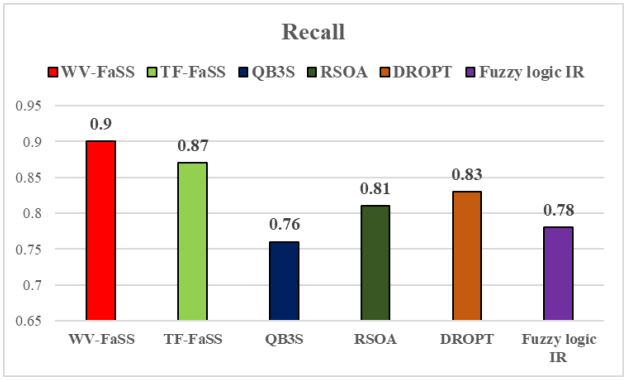

Fig. 11. Comparison of Recall Scores

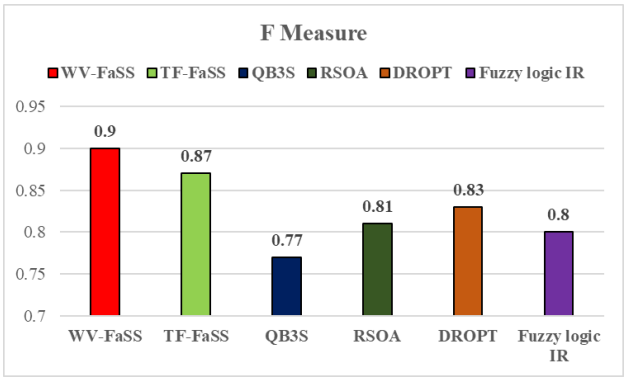

Fig. 12. Comparison of F1 Scores

Table 10. Analysis of Computation Time

Number of queries	Computation Time (s)					
	WV-FaSS	TF-FaSS	QB3S	RSOA	DROPT	Fuzzy Logic IR
10	13	16	21	29	18	28
20	29	34	38	41	48	52
30	45	53	59	61	58	77
40	68	89	91	98	109	98
50	81	109	111	128	137	147
60	101	121	129	157	168	185
70	119	148	153	189	208	218
80	134	167	176	211	239	245
90	158	194	199	238	257	269
100	185	209	212	259	287	297

Table 10 compares the computation time of the proposed WV-FaSS to that of other existing approaches such as TF-FaSS, QB3S, RSOA, DROPT, and Fuzzy Logic IR. The experimental procedure considers a variable number of query ranges. The queries in the range of 10 to 100 are considered from the SRS. When the number of queries is between 10 and 50, the computational time for the proposed WS-FaSS technique is between 13 and 84 seconds, whereas the existing TF-FaSS, QB3S, RSOA, DROPT, and Fuzzy Logic IR techniques take between 16 and 109 seconds, 29 to 128 seconds, 18 to 137 seconds, and 28 to 147 seconds, respectively. The computation time analysis reveals that when the number of queries is small (less than 50), the proposed WV-FaSS technique minimizes computation time significantly; however, when the number of queries is large (greater than 60), the proposed WV-FaSS technique achieves a large deviation in computation time and significantly outperformed the existing techniques.

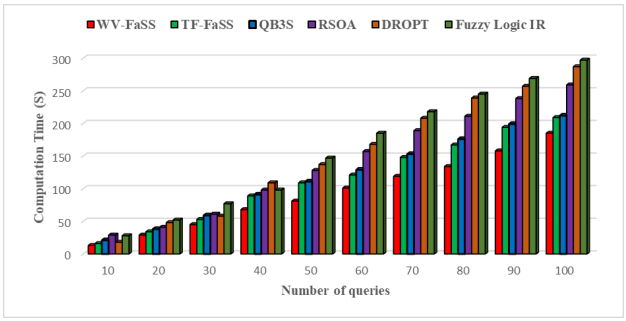

Fig. 13. Comparison of Computation Time

Fig. 13 describes the computation time comparison along with queries in the range of 50-100. From the analysis, it is observed that the lower computation time is attained for the WV-FaSS technique while compared with the existing techniques. On the whole, TF-FaSS outperforms well compared to DROPT, QB3S, RSOA, and Fuzzy Logic IR. However, 18.15% of computation time is minimized using the proposed WV-FaSS compared to TF-FaSS.

6 Threat to Validity

This section summarises the major constraints and threats to the validity of the experiments done. Although every effort has been made to ensure that the outcome was not impacted by adverse conditions, there are a few aspects to consider before recreating these experiments:

- The primary threat is that dataset annotation was performed manually, implying some degree of objectivity and reliability. To mitigate this danger, a deliberate process was used to establish the ground truth. A guideline for annotation is created and many trial runs are undertaken, followed by a reconciliation of any discrepancies. Finally, assessed the annotation's quality using inter-rater agreement metrics.
- Also, concerns about reliability pertain to the extent to which the data and analysis are dependent on the researchers. In theory, if another researcher repeats the same study later, the results should be identical. All findings in this study were derived by at least two researchers and then reviewed by at least three additional researchers. As a result, this threat has been diminished.
- The framework is intended to be independent of the context in which it is implemented. However, because it has not been tested in a multitude of environments, it is possible that some unique restrictions have not been taken into consideration. Precision, recall, and F1 metrics are utilized to assess the effectiveness of requirement extraction and classification approaches to reduce the threat.

To recapitulate, the researchers claim that the risks to the results' validity have been mitigated, though the inferences should not be applied to all businesses.

7 Conclusion

The research findings indicate that it is critical to properly identify data-intensive requirements in SRS to ensure the successful development of software-intensive projects. The novelty of this paper lies in the retrieval of transition requirements, specifically for DIAs. The retrieval of pertinent data enables the development of significant insights into business intelligence difficulties. Vectorizing requirements documents using word embeddings enables semantic analysis of the texts. Along with Word2Vec, using a fast similarity (k -NN) search, it retrieved the requirements independently as business requirements, stakeholder requirements, solution requirements, and transition requirements. Additionally, it evaluated the extracted documents' impact by comparing their performance to metrics derived from a comparison of the proposed WV-FaSS to state-of-the-art methodologies using benchmark datasets. Precision, recall, and F1 have values of 0.91, 0.90, and 0.90, respectively. As a result, retrieval of data-intensive requirements enables developers to more efficiently capture their initiatives by eliminating rework. The work is innovative since it does not compare the query document to all training documents. Rather than, an inverted index is utilized to discover documents that share vectors with the query vectors rapidly. Also, compared to existing strategies, the suggested WV-FaSS saves 18.15 percent of computing time.

References

1. Wang P. Eliciting big data requirement from big data itself: A task-directed approach. 6th International Workshop on Software Mining (Software Mining). 2017
2. Palomares C, Quer C, Franch X. Requirements reuse and requirement patterns: a state of the practice survey. Empirical Software Engineering. 2017; 22:2719–2762.
3. Robinson WN, Pawlowski SD, Volkov V. Requirements Interaction Management. ACM Computing Surveys (CSUR). 2003;35(2):132–190.
4. Meth H, Brhel M, Maedche A. The state of the art in automated requirements elicitation. Information and Software Technology. 2013; 55:1695–1709.
5. Pohl K. Rocky Nook, Inc; 2016.
6. Li C. Automatically classifying user requests in crowdsourcing requirements engineering. Journal of Systems and Software. 2018; 138:108–123.
7. Madhavji NH, Miranskyy A, Kontogiannis K. Big picture of big data software engineering: with example research challenges. IEEE/ACM 1st International Workshop on Big Data Software Engineering. 2015.
8. Sodagari E, Keyvanpour M. Challenges Classification of Software Requirements In- teraction Management Using Search-Based Methods. 5th International Conference on Web Research (ICWR). 2019.
9. Binkhonain M, Zhao L. A review of machine learning algorithms for identification and classification of non-functional requirements. Expert Systems with Applications: X. 2019; 1:100001–100001.
10. Merugu R, Ramesh SR, Chinnam. Automated cloud service-based quality requirement classification for software requirement specification. Evolutionary Intelligence. 2019; p. 1–6.
11. Hailes, Jarett. *Business Analysis Based on BABOK® Guide Version 3 Pocket Guide*. Van Haren, 2015.
12. Ferrari, Alessio, Giorgio Oronzo Spagnolo, and Stefania Gnesi. "Pure: A dataset of public requirements documents." *2017 IEEE 25th*

International Requirements Engineering Conference (RE). IEEE, 2017.
13. Hayes, Jane Huffman, Jared Payne, and Mallory Leppelmeier. "Toward Improved Artificial Intelligence in Requirements Engineering: Metadata for Tracing Datasets." *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019.
14. Gupta, Varun. "Requirement Engineering Challenges for Social Sector Software Development: Insights from Multiple Case Studies." *Digital Government: Research and Practice* 2.4 (2021): 1-13.
15. V. Vliet, Software Engineering: Principles and Practices. John Wiley & Sons, West Sussex, England, 2000.
16. N. Alhindawi, O. M. Al-Hazaimeh, R. Malkawi, and J. Alsakran, "A Topic Modeling Based Solution for Confirming and Assessing Software Documentation Quality," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 7, p. 7, 2016.
17. A. Aguilar, A. Zaldívar-Colado, C. Tripp-Barba, S. Misra, R.Bernal, and A. Ocegueda, "An Analysis of Techniques and Tools for Requirements Elicitation in Model-Driven Web Engineering Methods," in Computational Science and Its Applications -- ICCSA 2015: 15th International Conference, Banff, AB, Canada, June 22-25, 2015,
18. N. Alhindawi, J. Alsakran, A. Rodan, and H. Faris, "A Survey of Concepts Location Enhancement for Program Comprehension and Maintenance," Journal of Software Engineering and Applications, Vol.07No.05, p. 9, 2014.
19. Alsakran, N. Alhindawi, and L. Alnemer, "Parallel coordinates metrics for classification visualization," in 2016 7th International Conference on Information and Communication Systems (ICICS), 2016, pp. 7-12.
20. O. Meqdadi, N. Alhindawi, M. L. Collard, and J. I. Maletic, "Towards Understanding Large-Scale Adaptive Changes from Version Histories," in 2013 IEEE International Conference on Software Maintenance, 2013, pp. 416-419.
21. N. Alhindawi, N. Dragan, M. L. Collard, and J. I. Maletic, "Improving Feature Location by Enhancing Source Code with Stereotypes," in 2013 IEEE International Conference on Software Maintenance, 2013, pp. 300-309.
22. H. Alsawalqah, H. Faris, I. Aljarah, L. Alnemer, and N. Alhindawi, "Hybrid SMOTE-Ensemble Approach for Software Defect Prediction," in Software Engineering Trends and Techniques in Intelligent Systems, Cham, 2017, pp. 355-366.
23. L. Chung and J. C. P. Leite, "On Non-Functional Requirements in Software Engineering," in Conceptual Modeling: Foundations and Applications, T. B. Alexander, K. C. Vinay, G. Paolo, and S. Y. Eric, Eds., ed: Springer-Verlag, 2009, pp. 363-379.
24. R. Sindhgatta and S. Thonse, "Functional and non-functional requirements specification for enterprise applications," presented at the Proceedings of the 6th international conference on Product Focused Software Process Improvement, Oulu, Finland, 2005.
25. Slankas J, Williams L. Automated extraction of non-functional requirements in available documentation. In: 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE); 2013. p. 9–16.
26. Jindal R, Malhotra R, Jain A. Automated classification of security requirements. In: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI); 2016. p. 2027–2033.
27. Lu M, Liang P. Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. In the Proc. 21st International Conference on Evaluation and Assessment in Software Engineering; 2017. p. 344–353.
28. Kurtanovic´ Z, Maalej W. Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. In: IEEE 25th International Requirements Engineering Conference (RE); 2017. p. 490–495.
29. Deocadez R, Harrison R, Rodriguez D. Automatically Classifying Requirements from App Stores: A Preliminary Study. In: IEEE 25th International Requirements Engineering Conference Workshops (REW); 2017. p. 367–371.
30. Kashina M, Lenivtceva ID, Kopanitsa GD. Preprocessing of unstructured medical data: the impact of each preprocessing stage on classification. Procedia Computer Science. 2020; 178:284–290.
31. Uysal AK, Gunal S. The impact of preprocessing on text classification. Information Processing & Management. 2014;50(1):104–112.
32. Anandarajan M, Hill C, Nolan TT, Preprocessing. Practical Text Analytics. Springer; 2019. p. 45–59.
33. Liang H, Sun X, Sun Y, Gao Y. Text feature extraction based on deep learning: a review. EURASIP journal on wireless communications and networking. 2017;(1):1– 12.
34. Dzisevic R, Sesok D. Text Classification Using Different Feature Extraction Ap- proaches. 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream). 2019; p. 1–4.
35. Canedo ED, Mendes B. Software Requirements Classification Using Machine Learning Algorithms. Entropy.2020;22(9):1057–1057.
36. Qader WA, Ameen MM, Ahmed BI. An Overview of Bag of Words; Importance, Im- plementation, Applications, and Challenges. 2019 International Engineering Con- ference (IEC). 2019; p. 200–204.
37. Lu M, Liang P. Automatic classification of non-functional requirements from aug- mented app user reviews. Proceedings of the 21st International Conference on Eval- uation and Assessment in Software Engineering. 2017; p. 344–353.
38. Qaiser S, Ali R. Text mining: use of TF-IDF to examine the relevance of words to documents. International Journal of Computer Applications. 2018;181(1):25–29.
39. Bounabi M, Moutaouakil KE, Satori K. Text classification using Fuzzy TF-IDF and Machine Learning Models. Proceedings of the 4th International Conference on Big Data and Internet of Things. 2019; p. 1–6.
40. Lakshmi R, Baskar S. Novel term weighting schemes for document representation based on ranking of terms and Fuzzy logic with semantic relationship of terms. Expert Systems with Applications. 2019; 137:493–503.
41. Campos R, Mangaravite V, Pasquali A, Jorge A, Nunes C, Jatowt A. YAKE! Key- word extraction from single documents using multiple local features. Information Sciences. 2020; 509:257–289.
42. Kadhim A. Term weighting for feature extraction on Twitter: A comparison be- tween BM25 and TF-IDF. 2019 International Conference on Advanced Science and Engineering (ICOASE). 2019; p. 124–128.
43. Walkowiak T, Datko S, Maciejewski H. Bag-of-words, bag-of-topics and word-to- vec based subject classification of text documents in polish-a comparative study. In: International Conference on Dependability and Complex Systems. Springer; 2018. p. 526–535.
44. Haque MA, Rahman MA, Siddik MS. Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study. 2019 1st Inter- national Conference on Advances in Science, Engineering and Robotics Technology (ICASERT). 2019; p. 1–5.
45. Lima M, Valle V, Costa E, Lira F, Gadelha B. Software Engineering Repositories: Expanding the PROMISE Database. XXXIII Brazilian Symposium on Software Engineering. 2020; p. 427–436.
46. Canedo ED, Mendes B. Software Requirements Classification Using Machine Learning Algorithms. Entropy.2020;22(9):1057–1057.
47. Behera B, Kumaravelan G. Text document classification using fuzzy rough set based on robust nearest neighbor (FRS-RNN). Soft Computing. 2020; p. 1–9.
48. Raharja IMS, Siahaan DO. Classification of non-functional requirements using fuzzy similarity knn based on iso/iec 25010. 2019 12th International Conference on Information & Communication Technology and System (ICTS). 2019; p. 264–269.
49. Deocadez R, Harrison R, Rodriguez D. Automatically classifying requirements from app stores: A preliminary study. 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). 2017; p. 367–371
50. Alrumaih H, Mirza A, Alsalamah H. Domain ontology for requirements classification in requirements engineering context. IEEE Access. 2020; 8:89899–89908.
51. Deocadez R, Harrison R, Rodriguez D. Automatically classifying requirements from app stores: A preliminary study. 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). 2017; p. 367–371.
52. Sainani A, Anish PR, Joshi V, Ghaisas S. Extracting and Classifying Requirements from Software Engineering Contracts. 2020 IEEE 28th International Requirements Engineering Conference (RE). 2020; p. 147–157.
53. Jindal R, Malhotra R, Jain A. Automated classification of security requirements. 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2016; p. 2027–2033.

54. Perez-Verdejo JM, Sánchez-García AJ, Ocharán-Hernández JO. A Systematic Lit- erature Review on Machine Learning for Automated Requirements Classification. 2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT). 2020; p. 21–28.
55. Krippendorff, Klaus. "Computing Krippendorff's alpha-reliability." (2011).
56. Lim SL, Finkelstein. A, anticipating change in requirements engineering. In: Re- lating Software Requirements and Architectures. Springer; 2011. p. 17–34.
57. Virmani D, Taneja S. A text preprocessing approach for efficacious information retrieval. In: Smart Innovations in Communication and Computational Sciences. Springer; 2019. p. 13–22.
58. Sarkar D., Text Analytics with Python, Apress, Second Edition, 2016.
59. Srinivasa-Desikan. Packt Publishing Ltd;2019.
60. Ma, Long, and Yanqing Zhang. "Using Word2Vec to process big text data." *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015.
61. Amorim, Leonardo Afonso, et al. "A fast similarity search knn for textual datasets." *2018 Symposium on High Performance Computing Systems (WSCAD)*. IEEE, 2018.
62. Kadiyala, Akhil, and Ashok Kumar. "Applications of python to evaluate environmental data science problems." *Environmental Progress & Sustainable Energy* 36.6 (2017): 1580-1586.
63. Mia, Md Raihan, and Abu Sayed Md Latiful Hoque. "Question bank similarity searching system (qb3s) using nlp and information retrieval technique." *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE, 2019.
64. Rahul, Kumar. "Rider Optimization Algorithm (ROA): An optimization solution for engineering problem." *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.12 (2021): 3197-3201.
65. Agbele, Kehinde K., et al. "Algorithm for information retrieval optimization." *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2016.
66. Gupta, Yogesh, Ashish Saini, and A. K. Saxena. "A new fuzzy logic based ranking function for efficient information retrieval system." *Expert Systems with Applications* 42.3 (2015): 1223-1234.