Retrieval of Interactive Requirements of Data Intensive Applications using Random Forest Classifier

Renita Raymond[1], S Margret Anouncia[2,*]
[1,2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India.
E-mail: renita.r@vit.ac.in[1], smargretanouncia@vit.ac.in[2,*]
[*]Corresponding Author

Keywords: requirement classification, interaction-based requirements, data-intensive applications, random forest classifier

Received: January 4, 2021

Classifying requirements in data-intensive systems based on their interactions can assist the requirements engineering process in becoming more systematic and transparent, resulting in higher requirement compliance and software project completion. However, understanding the requirements centred on interactions with the system is particularly tough due to the increased complexity of big data. In most cases, awareness of interaction-based requirements is critical in moving forward with prediction and decision-making. As a result, the classification of interactive requirements plays a critical role in removing the difficulties from unclear requirements. Various approaches to effective requirement classification are being devised. However, due to inadequate requirement management reflecting the fast-changing organizational change, classification accuracy does not achieve its maximum potential. The best approach for reducing misclassification rate and retrieving interactive requirements for data-intensive systems would be to use Word Embedding and Random Forest Classifier retrieval mechanism, as none of the studies to date have emphasized it. It also assessed the impact by comparing the results to metrics derived from the Random Forest classifier's training on word count characteristics. The data set used to experiment with the classification, particularly for interaction-based needs, is unique to our work and has not been covered by any other studies to date. The researchers will benefit from this study as they will better understand the requirement classification process. With an F1 score of 0.91, precision of 0.89, and recall of 0.93, statistical analysis showed that Word Embedding followed by Random Forest Classifier produced a relatively high classification result to differentiate interactive requirements for data-intensive systems.

Povzetek: Z uporabo algoritma naključnih gozdov je izboljšana klasifikacija interaktivnih zahtev v podatkovno intenzivnih aplikacijah.

1 Introduction

Generally, Big Data is described as a massive chunk of unstructured and structured data making it difficult to process using traditional methods [1]. Data-Intensive Applications (DIA) help business organizations to drive predictive and informative decisions by analyzing this massive chunk of data. Big data software requirements discover business values. Therefore, to elicit software requirements for the DIA, it is necessary to outline the implication of the projects at an earlier stage [2]. Requirement Engineering (RE) assist in collaborating with various stakeholders and business analyst expertise in analytical thinking to perceive and comply with the value and priority of each requirement. According to statistics, RE was the source of 60% of software development faults. As a result, soliciting relevant requirements minimizes the risk of software-intensive projects and consequently improves quality [3].

Requirements are also iterative, dynamic, interactive, and never complete [4]. As most of the requirements written are in natural language, developers, analysts, and software architects always find it difficult to classify the requirements as it is time-consuming and error-prone manually. These tasks require expertise, training, experience, and domain knowledge [5]. By utilizing Natural Language Processing (NLP), developers can organize and structure the requirements to perform feature extraction, classification, speech recognition, etc. Appropriate classification of requirements from Software Requirement Specification (SRS) improves the quality of software-intensive products [6]. Nevertheless, the requirements engineering process for traditional and big data business intelligence systems share many commonalities, and it also differs in many aspects. A very clear sympathetic is necessary to understand and classify the interactive requirements for end-user applications [7]. In DIA, interactive requirements must be processed separately and classified accurately to improve the quality of requirements and reduce budget over-run. Techniques for automatically classifying the elicited interactive-based requirements into different classes are required [8]. According to Manal et al. [9], Machine Learning (ML) approaches for classifying requirements in requirement documents have produced better results than traditional natural language processing approaches. However, the systematic level of understanding is still lacking.

Similarly, various approaches are used to classify functional and non -functional requirements [10]. However, there was no automated tool to support the analysis and management of interactive-based requirements, leading to various consequences like budget overrun, quality and security issues, and customer dissatisfaction in DIA. Furthermore, as the vast amount of data generated is increasing significantly on the internet, it is formidable for the developers to categorize and extract meaningful information especially textual requirements from the SRS, due to their complex semantic meaning. A supervised machine learning technique is used for the classification of requirements. Based on the acquired knowledge from training, it is possible to categorize analogous documents into various classes. Nevertheless, it is a more challenging task when designing DIA as the corpus to be classified increases to million petabytes every day on the internet. As mentioned earlier, word embedding and an improved random forest algorithm help catalogue the interactive requirements from the SRS.

In the proposed framework, requirement feature extraction and requirement document classification are the two significant steps. In the first step, text features extracted are from the SRS documents using pre-processing. The extracted text features are represented as real-valued feature vectors in a predefined vector space using word embedding. In the next phase of classification, the converted feature vectors are categorized into four types, namely Input Requirement (IReq), Output Requirement (OReq), Transaction Requirement (TSReq), and Transformation Requirement (TFReq). IReq is the set of requirements from the environment required to produce a given level of outputs, OReq is the set requirements provisioned for the environment, TSReq is the set of requirements that are stable and filtered out. TFReq is the computation performed based on the requirement. Then, a query set is created with the help of keywords seen in the SRS. Non Metric Space Library (NMSLIB) creates indexing and retrieves the most similar documents according to the query set with a similarity score. Also, the performance is measured by training the corpus using the improved random forest classifier algorithm.

The remainder of the paper is structured as follows. Section 2 and 3 consist of related work and motivation. Section 4 explains the design and implementation of our retrieval of IREq, OReq, TFReq, and TSReq using similarity search and some background work associated with it, and Section 5 presents results. Finally, Section 6 consists of some conclusions along with the future scope.

2 Related work

RE is one of the essential aspects of research in the field of software engineering. Studies proclaim that failure to understand and classify requirements are the root cause for exceeding the allocated budget and time, leading to software system failure. They were manually classifying the requirements accordingly as FR and NFRs are difficult. Several researchers have stated that the requirements can be extracted and classified as FR and NFRs automatically from the natural language documents using various machine learning approaches and fuzzy techniques. Many techniques, especially for the classification of NFRs, have been devised and applied to various applications. Nevertheless, none of the methods addressed the classification of interaction-based requirements for data-intensive applications like banking, e-commerce, etc. This section outlines the various methods involved classification of requirements generally.

A software system's success depends significantly upon adherence to non-functional requirements because when it is being missed or ignored, significant issues arise. To address this issue, Slank et al. [13] proposed a tool-based approach, namely the NFR locator. This tool classifies and extracts the sentences in natural language texts into their respective NFR categories. Though the NFR locator helps the analyst effectively extract NFRs in available natural language documents through automated NLP, it works well only with texts. It cannot process images and tables in the unconstrained document present. Similarly, security-related issues must be considered with caution for completing software that meets the customer's needs. Text mining techniques and prediction models have been used to classify the security requirements [14].

In 2017, Liang et al. [15, 16] combined feature extraction and machine learning algorithms to classify user review requirements automatically and concluded that AUR -BoW with Bagging provides the best classification results. Requirements can also be classified as FR and NFRs accurately using semi-supervised and unsupervised machine learning algorithms.

A semi-Supervised classification technique can also be used to extract the FR and NFRs from the SRS automatically. Compared to supervised techniques, Semi-Supervised techniques provide better results because, in the latter one, only a minor amount of data needs to be labelled. In the former one, all the data set need to be labelled for classification. One such example is the app store, where the requirements present in the review from the app store are classified as functional and non-functional requirements using a self-labelling algorithm which is a part of the semi-supervised classification technique [17]. Semi-supervised classification methods help in classifying the requirements accordingly. Also, it will be enhanced with unsupervised learning techniques in the future.

2.1 Requirement pre-processing

SRS consists of incredibly massive data of all sorts, and they are heterogeneous by nature with inconsistent values. Pre-processing is a very crucial task that must be completed before the data is used for model training. Authors of [47, 49] alleged the main pre-processing stages as tokenization, stop words removal, error correction, normalization, and vectorization. Uysal et al. [48] evaluated the combination of pre-processing methods on two domains, namely e-mail and news, in two different languages. Results showed that choosing appropriate combinations of pre-processing tasks significantly improves classification accuracy depending on the domain and language studied. It is evident that pre-processing leads to better data sets that are clean and more manageable and must for any business organization to get meaningful insights.

2.2 Feature extraction

An SRS, modelled after business requirement specification, consists of all the requirements categorized into four types: IReq, OReq, TSReq, and TFReq. It is represented in vectors after pre-processing so that the machine learning algorithms train the corpus and classify it accordingly. The feature extraction process extracts the text features from the SRS documents using NLP pre-processing techniques by converting text into feature vectors.

Feature Extraction improves the accuracy of the learning algorithm as well as shortens the time. Selecting features from some effective ways like the vector space model reduces feature space dimensions [18]. Feature extraction algorithms like Term Frequency – Inverse Document Frequency (TF-IDF), Bag of Words (BoW), and Word2Vec calculate the weights of the words in the text by initiating a feature vector of the text using a predefined keyword set [19]. This section includes various feature extraction techniques used to extract the features and their limitations.

One hot encoding is the first count-based embedding technique that converts the text into a vector by constructing a vocabulary. However, it cannot capture any contextual information due to its inefficient memory requirement [28].

BoW is one of the most common and effective features extraction techniques because of its simplicity and performance. In BoW, assuming words are independent of each other, texts are represented as a bag of words by recording the number of occurrences of each instance or word in a bag irrespective of their order or grammar. However, it leads to a high sparse and dimensional feature vector due to its non-zero dimensions and large vocabulary size [21][22]. Using Bow, all the features will have a value, and it gives equal weightage to all the features in the documents. Additionally, recurrently appearing features direct the model rather than the importance of the features in the document which TFIDF is solving.

Qaisier et al. [23] say that TFIDF is calculated by multiplying both the term frequency and Inverse document frequencies. Terms with high TF-IDF weight are considered to be more important rather than terms with lesser TFIDF scores.

However, TF-IDF is the most well-known and used formula to produce a vectors descriptor that developed to have several normalized forms it has certain limitations. TFIDF does not care about the position of a term in the text, its semantics and co-occurrences with other texts in the documents. In 2019, an extended form of Fuzzy based TF-IDF (FTF-IDF) is introduced to overcome the limitations of TF-IDF. FTF-IDF is a vector representation, where the components of the TF-IDF are presented as inputs to the Fuzzy Inference System (FIS). Weight terms are generated as crisp outputs after the defuzzification step. FTF-IDF provides semantic meanings to the words in the documents [24]. It does not look into the co-occurrences of other texts in the documents.

Later on in the same year, Lakshmi et al. [25] proposed term weighting schemes 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 provides better clustering performance in terms of accuracy, recall, and F1 measure compared to word count, Term Frequency-Inverse Document Frequency (TF-IDF), Term Frequency-Inverse Corpus Frequency (TF- ICF), Multi-Aspect TF (MATF), BM25, and BM25F. Yet, it does not focus on the syntactic of the sentences in the documents.

Also, Ricardo et al. [20] initiated YAKE depending only on statistical text features and not on a trained large corpus. It is adapted to different languages and scalable to documents of any length. However, it cannot tackle manually assigned keywords when not found in the text.

Okapi BM25 is a ranking function used to estimate the relevance of documents to a given search query regardless of their proximity within the document. The authors of [27] made a comparative analysis using the Twitter data set and proved that TF-IDF is the best feature extraction technique compared to BM25 with an F1 measure of 89.77. BM25 is not suitable for large corpus.

The authors of [26] state that the selection of the weighting technique is not essential because the weighting process is just a linear transformation of feature vectors. Therefore, researchers can use any one of the text feature extraction techniques or the combination of various techniques based on their project requirement, as every method has its pros and cons.

2.3 Requirement classification

Requirements need to be defined, organized, and clearly understood by the stakeholders and the project members involved in developing the system. Classifying the requirements helps us define and organize the work because sometimes, compared to functional requirements, designing a system concerning non-functional requirements should be focused on a lot. It takes up large portions of the schedule and is filled with knotty problems. A part of requirement engineering, i.e., classification of requirements appropriately, is essential because it is the base for any software to be developed. Requirement classification done manually is a time-consuming task, and it is error-prone. Henceforth, an automatic classification of requirements must minimize rework and make the software easier to use and understand. This section consists of various classification techniques suggested by the researchers to classify the requirements automatically.

In 2019, Rahman et al. [30] extracted NFR from the SRS document using various machine learning techniques to meet customer expectations completely. Based on the statistical analysis, it is revealed that the SVM classifier achieves the best results with a precision of 0.66, recall of 0.61, and accuracy of 0.76. The experiments were conducted with the well-known PROMISE dataset, which has the characteristics of being unbalanced in FRs and NFRs. Lima et al. [31] expanded the PROMISE dataset, forming the PROMISE_exp repository.

Again, Edna et al. [29] showed a comparative analysis of various machine learning algorithms like Support Vector Machine (SVM), KNN (K Nearest Neighbour), Decision Tree, Multinomial Naive Bayes (MNB), and Logistic Regression (LR) to determine which algorithm fits better to classify the requirements automatically using PROMISE_exp. The results reveal that the combination of TF-IDF and LR has the best performance measures with an F-measure of 91% on the binary classification, 74% in 11 granularity classification, and 78% on the 12-granularity classification.

Before conducting any experimental analysis, researchers must verify whether the dataset being used is balanced or unbalanced. Studies have shown that an unbalanced dataset leads to poor automatic classification of requirements.

Fuzzy Rough Set (FRS) is a powerful mathematical tool to deal with uncertain data. So, Behera et al. [33] proposed a Fuzzy Rough Set based on Robust Nearest Neighbor (FRS-RNN) to document classification. A modified CNN is used to extract the features from the documents, and later on, using FRS-RNN, documents are classified. It outperforms all the classification models like SVM, Naive Bayes, DNN, and CNN. However, the hyperparameter tuning of FRS-RNN consumes more time than conventional machine learning algorithms.

An NFR sentence can be classified into more than one class. In 2019, Fuzzy Similarity KNN (FSKNN) was suggested for multi-label classification of requirements based on ISO/lEe 25010. In this paper, the fuzzy similarity measure approach is used to calculate the similarity between the terms, documents and a training pattern is obtained. The search set obtained from the training data is used to find the K nearest neighbor. A test document will be labelled into a specific category using a maximum a posteriori (MAP) estimate [35].

Similarly, to classify the FR and NFR contained in the reviews within the APP store, a semi-supervised classification technique was used. The self-labelling algorithm appropriately assigns labels to the collected unlabelled data and also classifies unseen future reviews. However, the results are not empirically evaluated [36].

Semantic information plays a significant role in the area of RE. Software developers use effective requirement classification techniques to produce semantic-based SRS of higher quality. A Requirement Classification Ontology (RCO) is initiated for sharing and describing the different classifications of requirements. It is used as a tool to confirm the RE process's semantic correctness, thereby ensuring consistency between the requirements [38].

Various studies [32][34][37] reveal that machine learning techniques play a significant role in classifying the requirements as FRs, NFRs, quality requirements, security requirements, legal requirements, etc., compared to fuzzy rule mechanisms.

However, from the related work, it is evident that no research has been carried out to address the challenges faced in extracting the interaction-based requirements nor sets the standards for categorizing the requirements based on their interactions for designing DIAs.

3 Motivation

It is inferred that categorizing the requirements according to their type of interactions will create transparency in the RE process, thereby promoting requirement fulfilment and completing software-intensive projects based on the study carried out. Considering the usefulness of the technology in software requirement classification, a new framework is designed to classify the interactive-based requirements. Limiting the requirements to interactions, in particular, can focus on what the DIA developers care about while allowing the engineers to bring all their knowledge and creativity to bear on the means for achieving it. Distinguishing interactive requirements from other requirements is very important because there are usually much more difficult challenges to design and test DIAs. Manuel et al. [46] conducted a survey in 2020, which reveals that the most recurrent classification algorithms featured on the identified studies are Naive Bayes, K Nearest Neighbor, J48, and Natural Language Processing algorithms. Also, the most used training datasets are academic databases and collected user reviews. Finally, it was concluded that most of the studies focus on classifying FRs and NFRs. None of the studies revealed the interest in classifying interactive requirements, especially for software-intensive projects.

4 Proposed methodology

Given the extraction of interactive requirements as a prime focus, the framework is designed with the following phases,

- ➢ Requirement Elicitation
- ➢ Requirements (Text) Pre-Processing
- ➢ Features Extraction
- ➢ Requirement Discovery
- ➢ Requirement Classification

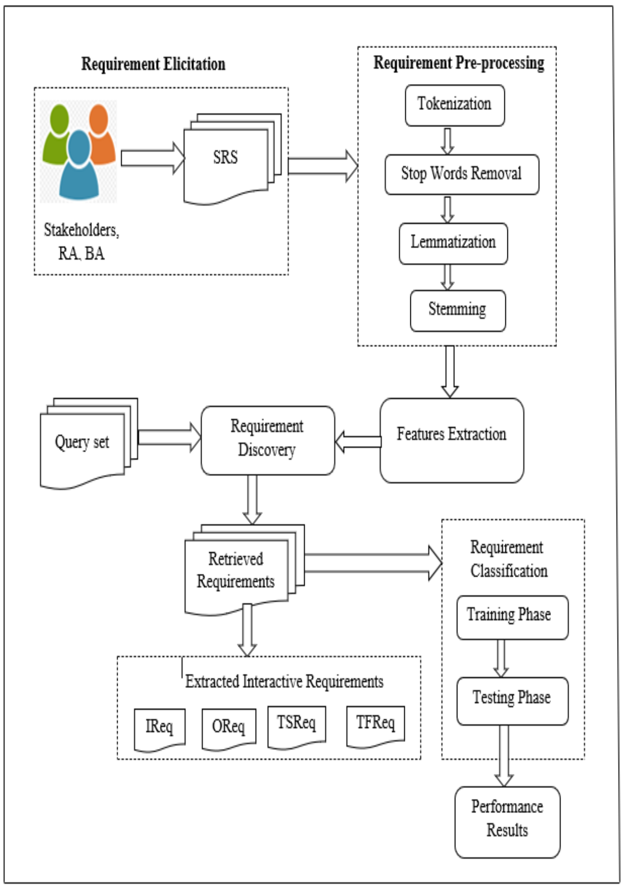
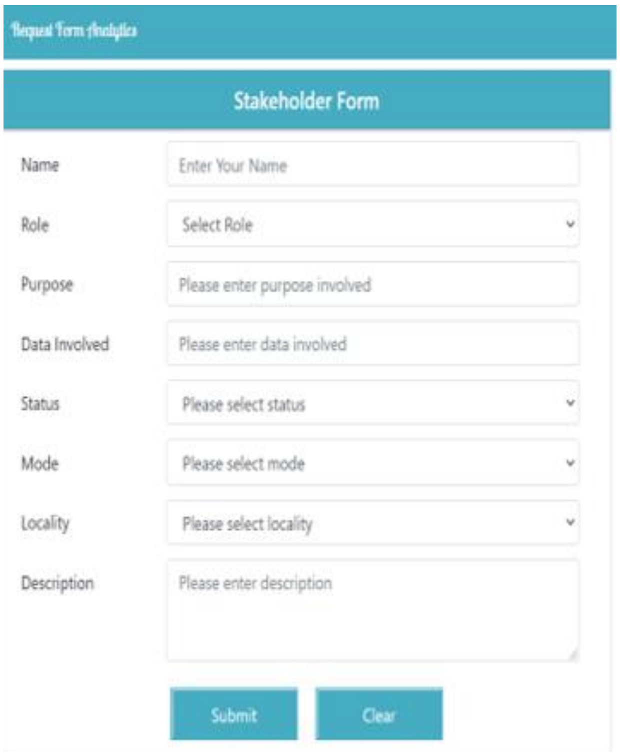


Figure 1: Framework for extracting interactive requirements

4.1 Requirement elicitation

In the RE phase, requirement elicitation discovers the requirements for developing software-intensive projects from the users, customers, and other stakeholders. The requirements of DIAs should be discovered in the initial stage of the software life cycle itself. Conventional RE processes are incapable of fulfilling the needs of the organization mainly for two reasons. Firstly, it focuses primarily on generic user requirements, and it does not provide any meaningful insights about the features generated from big data's leading to a better business intelligence solution. Secondly, the vast amount of data generated daily by various systems leads to increased demand for consumption at various levels. Therefore, in the process of requirement elicitation in DIAs, even business analysts are also involved in the discussion to provide business intelligence solutions to the organizations.

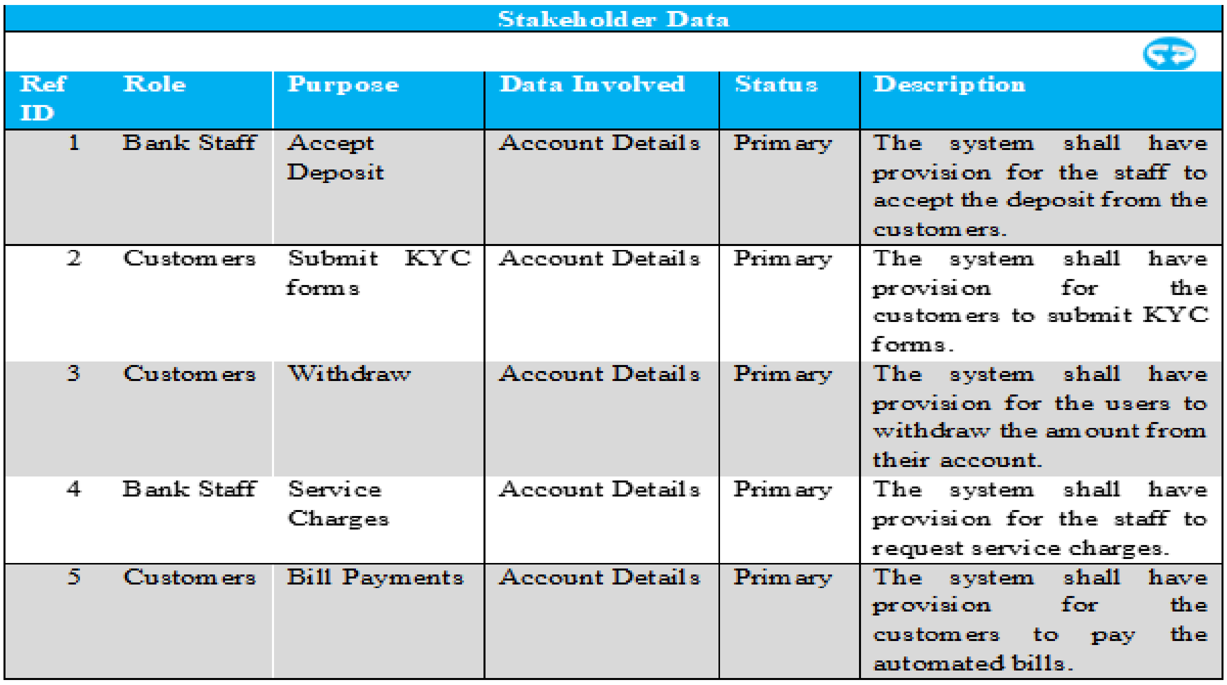
In the first phase of the framework, a form has been designed to gather the requirements from various stakeholders. The dataset created for this paper is based on the banking application. The stakeholders of the banking domain are customers, bankers, investors, regulators, RBI, etc. Requirements are gathered from the stakeholders and documented initially. The stakeholder form created for gathering the requirements is shown in figure 2. The form includes various details of a requirement like the name of the stakeholder, the role of the stakeholder (i.e., customer, staff, BoD, Investors, Regulators, etc.), purpose, data required for the particular requirement, the status of the stakeholder, either primary or secondary stakeholder, mode of interaction when entering the requirement, locality and the description of the requirement.


Request Form Analytics

Stakeholder Form

Name	Enter Your Name
Role	Select Role
Purpose	Please enter purpose involved
Data Involved	Please enter data involved
Status	Please select status
Mode	Please select mode
Locality	Please select locality
Description	Please enter description

Figure 2: Stakeholder form



Stakeholder Data

Ref ID	Role	Purpose	Data Involved	Status	Description
1	Bank Staff	Accept Deposit	Account Details	Primary	The system shall have provision for the staff to accept the deposit from the customers.
2	Customers	Submit KYC forms	Account Details	Primary	The system shall have provision for the customers to submit KYC forms.
3	Customers	Withdraw	Account Details	Primary	The system shall have provision for the users to withdraw the amount from their account.
4	Bank Staff	Service Charges	Account Details	Primary	The system shall have provision for the staff to request service charges.
5	Customers	Bill Payments	Account Details	Primary	The system shall have provision for customers to pay the automated bills.

Figure 3: Stakeholder data

Requirement Analyst analyses the difference between what the customers need, validates, and documents the need of the project stakeholders. During the analysis phase, the analyst identifies the gathered requirements type documented using stakeholder form as either stable or volatile requirement concerning their priority and feasibility. Requirement types can be divided into two type's stable and volatile requirements [39].

Stable Requirement – otherwise called enduring requirements are the requirements derived from the organization's core activity and directly related to the system's domain. Here, in the banking domain, requirements concerned with customers, bankers who do not change on time are considered. For example, 'The system shall have provision for the customers to deposit amount in the account', 'The system shall have provision for the staff to get the customer details when opening an account.'

Volatile Requirements – requirements that are likely to change after the system becomes operational are considered volatile requirements. Requirements related to policies framed by the Board of Directors, Investors, RBI are included in it. Such type of requirements falls into four categories as follows.

Mutable – change in requirements concerning changes triggered in the organization's environment is included in it. E.g., 'The system shall have provision for the staffs to initiate the customers to set transaction limit for the transactions'.

Emergent – requirements that emerge when the system is being developed and implemented are included in it. For example, 'The system shall have the staff's provision to collect the debt loan from the customers when it is not being repaid after giving prior notice'.

Consequential – requirements that result from the introduction of the computer system are known as consequential. For example, 'The system shall have provision for the staff to link the customers' account details with aadhar card'.

Compatibility – requirements that depend on other equipment or processes are included in it. E.g., 'The bank will have many ATMs, and the new software shall provide all the ATMs' functionality'.

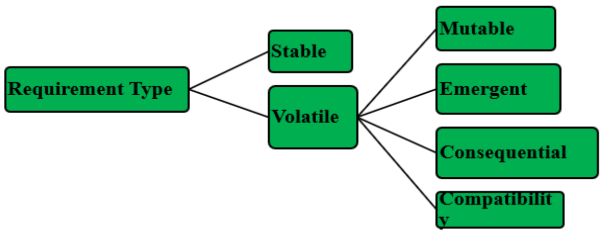


Figure 4: Requirement types

A separate keyword list is created and catalogued, as shown in Table 1. The Interaction Type column represents the four interaction types as Input, Output, Transformation, and Transaction. Various keywords related to the interaction types are listed in the Keywords column. Keywords present in the description column of the stakeholder data depicted in Figure 3 are matched with the Interactive Requirement Keyword Catalogue. The requirements are classified as Input, Output, Transaction, and Transformation automatically concerning their requirement type, priority, and feasibility. Any specific requirements needed for the corresponding requirements are also recorded and finally documented, as depicted in figure 6.

Table 1: Interactive requirement keyword catalogue

S. No	**Interaction Type**	**Keywords**	**Total No of Keywords**
1	Input (IReq)	Get, Login, set transaction limit, check, Request, Raise, Write, Complete, set, enter, receive, open, verify, ensure, submit, evaluate, select, monitor, maintain, maintain Debt	20
2	Output (OReq)	view, display, print, provide, canvassing, conduct, sanction, respond, issue, appoint, take, review, limit, observe, obtain	15
3	Transformation (TFReq)	Deposit, invest, pay, recharge, withdraw, transfer, add, accept, update, exchange, set policy, set priorities, link account	13
4	Transaction (TSReq)	Calculate EMI, Packaging and rolling, quarterly, Filter, year, lock, authorization, evaluate	8

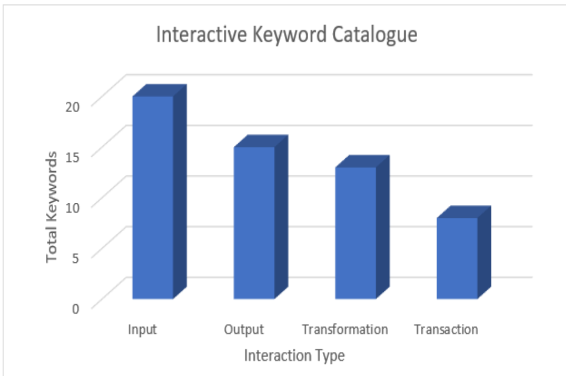


Figure 5: Distribution of interactive requirement type keywords catalogue

Table 1 and Figure 5 show the distribution of keywords concerning their interaction types. Out of 56 keywords, Input consists of 20, the output consists of 15, Transformation consists of 13, and transaction consists of 8 keywords.

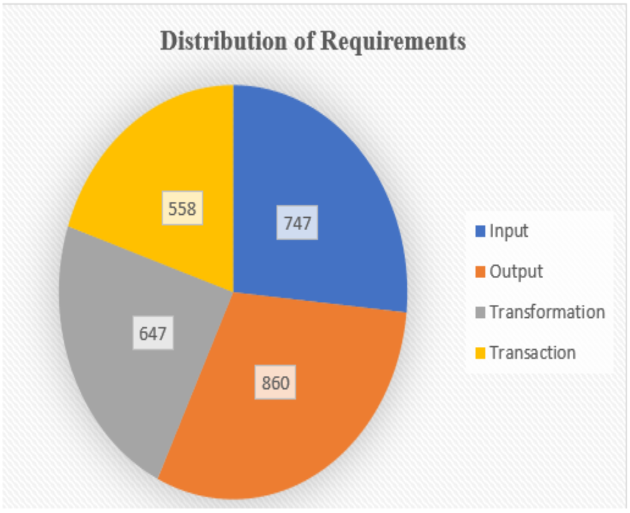


Figure 7: Distribution of requirements per category

The corpus created consists of 2812 requirement instances finally after the approval of the requirement analyst. The distribution of the requirement instances is shown in figure 7. IReq consists of 747 instances, OReq consists of 860 instances, TFReq consists of 647 instances, and TSReq consists of 558 instances.

4.2 Requirement pre-processing

Requirement Pre-Processing is the second stage of the classification process. It directly improves the model's performance by removing the noise or unclear data extracted from different sources. Series of steps are followed to standardize textual data into a form that would be taken up as an input to analytics systems and applications. To categorize the requirement documents, there are various pre-processing techniques like stop words removal, tokenization, stemming, lemmatization, etc. Text from the SRS is broken into meaningful tokens. After converting into meaningful tokens, predefined stop words are removed. Occasionally, even the stop words can be user-defined based on their respective applications. Removing such words from the corpus reduces the dimensionality of the term space, thereby increasing the model's performance. Later on, stemming is done to identify the root of a token in the corpus. This process removes the various suffixes, reducing the corpus tokens even more to save time and memory space. Finally, lemmatization considers the morphological analysis of the tokens or words, thereby decreasing the noise and speeding up the user's task [40, 41].

Table 2: Corpus before pre-processing

RID	Description	Interaction Type
1	The system shall have provision for the users to login with authentication	Input
2	The system shall have provision to accept the deposit money of the customers	Transformation
3	The system shall have provision to request customers to maintain sufficient balance	Input
4	The system shall have provision to open an account for the customers	Transformation
5	The system shall have provision to submit customers KYC forms	Input
6	The system shall have provision to submit income statement of the customers	Input
7	The system shall have provision to set transaction limit for the transactions by the customers	Input
8	The system shall have provision for the customers to invest shares	Transformation
9	The system shall have provision for the users to pay automated bill payments	Transformation
10	The system shall have provision for the users to pay taxes	Transformation
11	The system shall have provision for the users to recharge the data card	Transformation
12	The system shall have provision for the customers to pay for travel through UPI	Transformation
13	The system shall have provision for the users to pay due (loan)	Transformation
14	The system shall have provision for the users to pay service charges	Transformation
15	The system shall have provision to for the users to set the ATM, Mobile Pin, Net Banking transaction pin	Input
16	The system shall have provision for the customers to calculate EMI for loan	Transaction
17	The system shall have provision for the customers to check the account balance of their account	Input
18	The system shall have provision for the users to withdraw the amount from their account	Transformation
19	The system shall have provision for the customers to view their weekly, monthly transaction details	Output
20	The system shall have provision for the customers to submit their personal details	Input

All requirements in the corpus have gone through a pre-processing step. Table 2 shows the requirements in the corpus before the pre-processing steps. In this paper, Spacy, a free, open-source library for NLP is being used to process and understand 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 [42]. For example, Table 2 RID 1: "The system shall have provision for the users to login with authentication" has been changed to "['user', 'login', 'authentication']" as shown in RID 1 of Table 3.

Table 3: Corpus after Text pre-processing

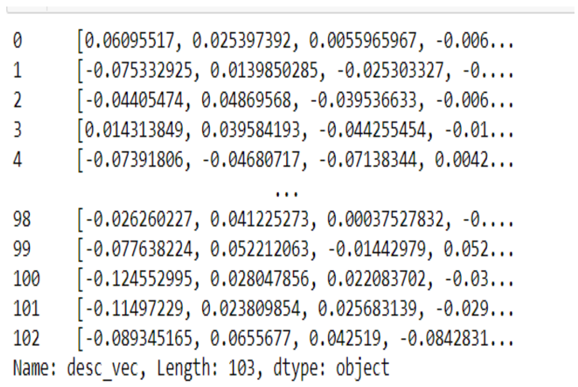
RID	Description	Interaction Type	Tokens
1	The system shall have provision for the users to login with authentication	Input	['user', 'login', 'authentication']
2	The system shall have provision to accept the deposit money of the customers	Transformation	['accept', 'deposit', 'money', 'customer']
3	The system shall have provision to request customers to maintain sufficient balance	Input	['request', 'customer', 'maintain', 'sufficient', 'balance']
4	The system shall have provision to open an account for the customers	Transformation	['open', 'account', 'customer']
5	The system shall have provision to submit customers KYC forms	Input	['submit', 'customer', 'kyc', 'form']
6	The system shall have provision to submit	Input	['submit', 'income',

RI D	Description	Interaction Type	Tokens
	income statement of the customers		'statement', 'customer']
7	The system shall have provision to set transaction limit for the transactions by the customers	Input	['set', 'transaction', 'limit', 'transaction', 'customer']
8	The system shall have provision for the customers to invest shares	Transformation	['customer', 'invest', 'share']
9	The system shall have provision for the users to pay automated bill payments	Transformation	['user', 'pay', 'automated', 'bill', 'payment']
10	The system shall have provision for the users to pay taxes	Transformation	['user', 'pay', 'tax']
11	The system shall have provision for the users to recharge the data card	Transformation	['user', 'recharge', 'data', 'card']
12	The system shall have provision for the customers to pay for travel through UPI	Transformation	['customer', 'pay', 'travel', 'upi']
13	The system shall have provision for the users to pay due (loan)	Transformation	['user', 'pay', 'due', 'loan']
14	The system shall have provision for the users to pay service charges	Transformation	['user', 'pay', 'service', 'charge']
15	The system shall have provision to for the users to set the ATM, Mobile Pin, Net Banking transaction pin	Input	['user', 'set', 'atm', 'mobile', 'pin', 'net', 'banking', 'transaction', 'pin']
16	The system shall have provision for the customers to calculate EMI for loan	Transaction	['customer', 'calculate', 'emi', 'loan']
17	The system shall have provision for the customers to check the account balance of their account	Input	['customer', 'check', 'account', 'balance', 'account']
18	The system shall have provision for the users to withdraw the amount from their account	Transformation	['user', 'withdraw', 'amount', 'account']
19	The system shall have provision for the customers to view their weekly, monthly transaction details	Output	['customer', 'view', 'weekly', 'monthly', 'transaction', 'detail']
20	The system shall have provision for the customers to submit their personal details	Input	['customer', 'submit', 'personal', 'detail']

The above table shows the corpus after wrangling, cleaning up, and standardizing the textual requirements into a form (i.e., tokens) taken up as an input for the feature extraction process.

4.3 Feature extraction

In this stage, the pre-processed corpus is converted into numerical features representing the information contained in the requirements usable for machine learning. As the actual text is highly dimensional and unstructured, every unique word or token is seen as a separate dimension, making it challenging to apply classification algorithms. Word2Vec [43], developed by Tomas et al., takes as its input a large corpus of tokens obtained from the normalization process producing a vector space for unique tokens. Words in the vector space that share familiar contexts in the corpus are located close to one another in the space. The word vectors obtained for the corpus is shown in figure 8.



```
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
```

Figure 8: Sample word vectors created using Spacy toolkit.

In the above figure, Spacy [42] parses entire blocks of text and seamlessly assigns word vectors from the loaded models. Word2vec improves the quality of features by considering contextual semantics of words in a text, hence improving machine learning and requirement classification accuracy.

4.4 Requirement discovery

Requirement discovery is the process of identifying the interactive requirements IReq, OReq, TFReq, and TSReq needed to design software-intensive projects respectively based on the query set created. It is the understanding of how such interactive requirements are formed internally and externally. Query set created consisting of keywords as shown in Figure 9 should be meaningful to the humans, and it should provide enough diverse results in retrieving the documents. These keywords generalize the features of the corresponding requirements, and many diverse compositions can be found by retrieving them.

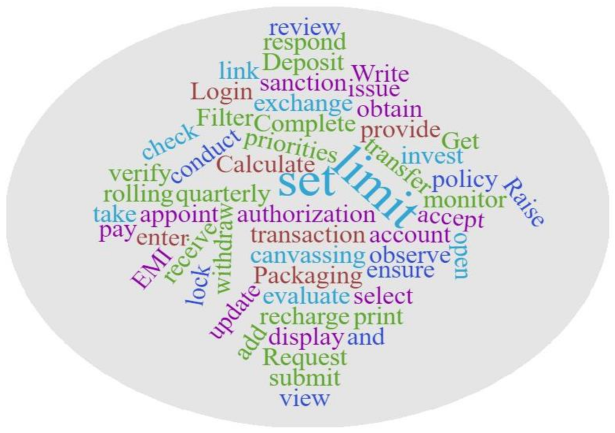
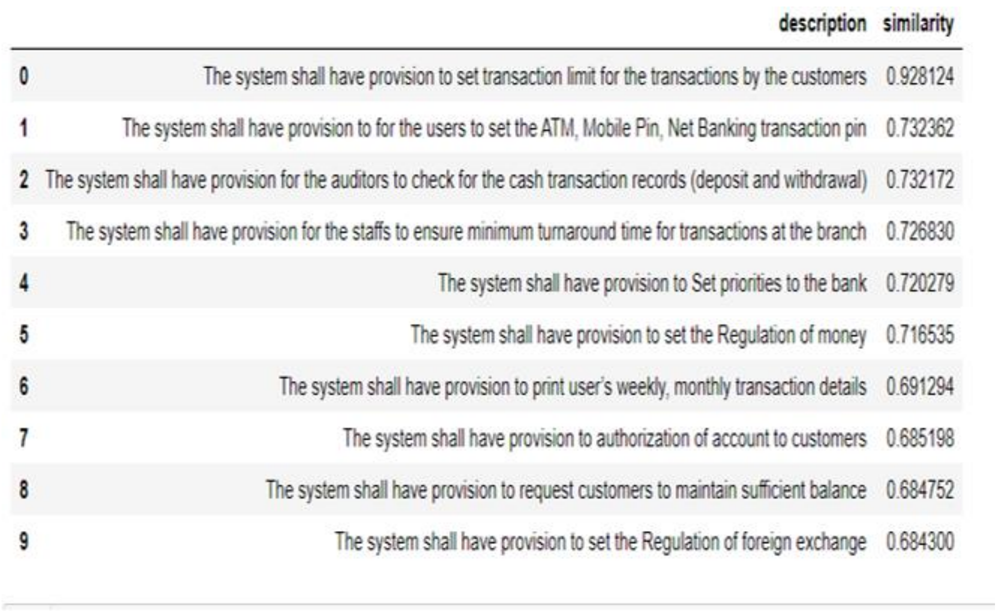


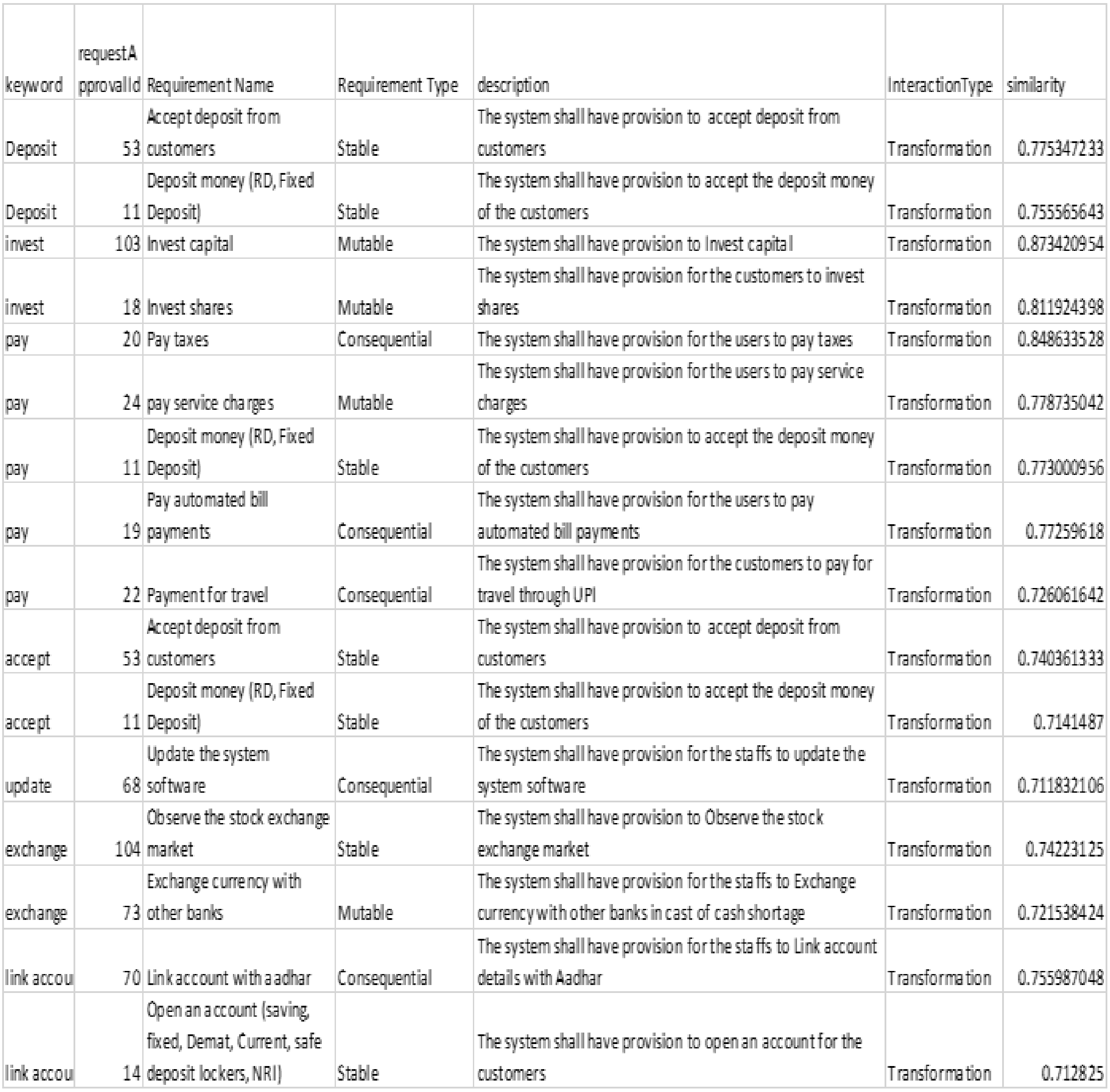
Figure 9: Word Cloud of the query set (keywords)

Features extracted from the Word2Vec are also passed as an input to the requirement discovery phase. A similarity measure is a metric used to measure the similarity between the features present in the corpus, irrespective of their sizes. This paper considers metric spaces and non-metric spaces because the non-metric similarity provides robustness, locality, and comfort in modelling. A non-metric is a function that does not satisfy some or all the properties of a metric. It includes context-dependent similarity functions and dynamic similarity functions as well. The non-Metric Space Library (NSMLIB) [44] is an efficient and extendable cross-platform similarity search library and a toolkit to evaluate similarity search methods. It is a library for fast similarity K Nearest Neighbour (k-NN) search. In this phase of extraction of interactive requirements based on the keywords present in the query set, NMSLIB is used as it is the first tool to support non-metric space searching. The principal concern is to provide a solution to a query by retrieving a subset of requirements from the corpus sufficiently similar to the query q.



	description	similarity
0	The system shall have provision to set transaction limit for the transactions by the customers	0.928124
1	The system shall have provision to for the users to set the ATM, Mobile Pin, Net Banking transaction pin	0.732362
2	The system shall have provision for the auditors to check for the cash transaction records (deposit and withdrawal)	0.732172
3	The system shall have provision for the staffs to ensure minimum turnaround time for transactions at the branch	0.726830
4	The system shall have provision to Set priorities to the bank	0.720279
5	The system shall have provision to set the Regulation of money	0.716535
6	The system shall have provision to print user's weekly, monthly transaction details	0.691294
7	The system shall have provision to authorization of account to customers	0.685198
8	The system shall have provision to request customers to maintain sufficient balance	0.684752
9	The system shall have provision to set the Regulation of foreign exchange	0.684300

Figure 10: Top 10 similar requirements retrieved for the query 'set transaction limit.'



keyword	requestApprovalId	Requirement Name	Requirement Type	description	InteractionType	similarity
Deposit	53	Accept deposit from customers	Stable	The system shall have provision to accept deposit from customers	Transformation	0.775347233
Deposit	11	Deposit money (RD, Fixed Deposit)	Stable	The system shall have provision to accept the deposit money of the customers	Transformation	0.755565643
invest	103	Invest capital	Mutable	The system shall have provision to Invest capital	Transformation	0.873420954
invest	18	Invest shares	Mutable	The system shall have provision for the customers to invest shares	Transformation	0.811924398
pay	20	Pay taxes	Consequential	The system shall have provision for the users to pay taxes	Transformation	0.848633528
pay	24	pay service charges	Mutable	The system shall have provision for the users to pay service charges	Transformation	0.778735042
pay	11	Deposit money (RD, Fixed Deposit)	Stable	The system shall have provision to accept the deposit money of the customers	Transformation	0.773000956
pay	19	Pay automated bill payments	Consequential	The system shall have provision for the users to pay automated bill payments	Transformation	0.77259618
pay	22	Payment for travel	Consequential	The system shall have provision for the customers to pay for travel through UPI	Transformation	0.726061642
accept	53	Accept deposit from customers	Stable	The system shall have provision to accept deposit from customers	Transformation	0.740361333
accept	11	Deposit money (RD, Fixed Deposit)	Stable	The system shall have provision to accept the deposit money of the customers	Transformation	0.7141487
update	68	Update the system software	Consequential	The system shall have provision for the staffs to update the system software	Transformation	0.711832106
exchange	104	Observe the stock exchange market	Stable	The system shall have provision to Observe the stock exchange market	Transformation	0.74223125
exchange	73	Exchange currency with other banks	Mutable	The system shall have provision for the staffs to Exchange currency with other banks in cast of cash shortage	Transformation	0.721538424
link accou	70	Link account with aadhar	Consequential	The system shall have provision for the staffs to Link account details with Aadhar	Transformation	0.755987048
link accou	14	Open an account (saving, fixed, Demat, Current, safe deposit lockers, NRI)	Stable	The system shall have provision to open an account for the customers	Transformation	0.712825

Figure 11: Retrieval of transformation requirements based on the query set

For example, the above figure shows that the top (k =10) nearest neighbours with a similarity score is displayed for the query' Set Transaction Limit'. NMSLIB uses the k-Nearest Neighbors (k-NN) algorithm for performing similarity search as it is prevalent, and the elements in the corpus are represented as vectors. With the help of NMSLIB, k-NN enables high scale, low latency nearest neighbor search on billions of documents across thousands of dimensions with the same ease.

The above figure illustrates the retrieval of interactive requirements, especially transformation requirements based on the query with a similarity score. A high degree of similarity score implies a high probability of retrieving the documents concerning the query accurately.

Therefore, requirements concerning their interactions are retrieved accurately and efficiently with the help of a fast similarity search (k-NN) NMSLIB.

Table 4: Sample of requirements retrieved for specific queries

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

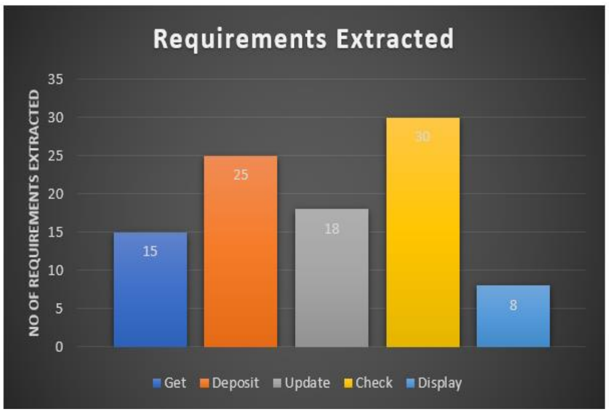


Figure 12: Sample of requirements retrieved concerning specific keywords

Table 4 and figure 12 depicts the sample of requirements retrieved with respect to specific keywords.

4.5 Requirement classification

Word embedding produced using Word2Vec is used to train the machine learning and classification algorithm to improve the interactive requirement classification accuracy based on the context and semantic relationship between words. Our approach uses a Random Forest as the training set fed into the algorithm entails labels. It contains several decision trees on various subsets of the given dataset and takes the majority votes to improve the predictive accuracy of that dataset [45]. The experimental analysis in section V reveals that it requires less training time than other algorithms and produces a high accuracy output even for the large data sets efficiently. Most industries consider the usage of Random Forest as it combines multiple classifiers to solve a complex problem, thereby maintaining accuracy even when the dataset is imbalanced. The corpus is split into a training dataset and testing data set in the ratio of 70:30. 70 % of the dataset goes into the training set, and the remaining 30% goes into the testing dataset. After splitting, the training set is trained using the RF model, and predictions are performed on the testing set. First, N random records with features from the training set are chosen, and secondly, a decision tree is build based on the N records. The parameter n_estimators decide the number of trees the RF needs, and the steps are repeated. A Decision Tree (DT) has low bias and high variance, prone to many errors when new test data arrives. Therefore, RF uses multiple DTs and row sampling and feature sampling concerning majority votes in the DTs. This way, high variance gets converted into low variance because using row sampling and feature sampling records to DT gets well trained concerning specific records. Evaluation metrics like precision, recall, and F1 measures are used to evaluate the classifier's performance.

4.6 Proposed algorithm for extraction of interactive requirements

The flow of the proposed methodology is as follows.

```
Algorithm: Extraction of Interactive Requirements
Input: let f represent the stakeholder form, SRS be the Software Requirement Specification, i be the i^th requirement in SRS
Output: let IReq, OReq, TFReq, and TSReq represent the Input Requirement, Output Requirement, Transformation Requirement, Transaction Requirement, respectively.
Data: Testing 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 back 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 ($S_i$, SimS)
Let $S_i$ be the tokens in corpus
word2vec model()
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
    Train the model
    Find the similarity score (SimS) for $s_i$
    Return SimS for the vectors $S_i$ in the corpus
endfor
Function Query Processing (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)
if $QS_i$ = $S_i$ in Corpus
    Retrieve the documents ($RD_i$) with SimS
else
    Return no match
Assign ExD ← $RD_i$
```

```
Function Classification
To generate k classifiers
Split the ExD in the ratio of 80:20 as 80 % training data and 20% testing data
foreach i= 1 to k do
    Sample the training data ExD
    $ExD_i \leftarrow ExD$
    Create a root node $RN_i$ containing $ExD_i$
    BuildTree ($RN_i$)
endfor
BuildTree (RN)
if RN consists of only one instance, then
    Return
else
    Select the features F randomly in RN
    Select F with the highest information gain to split on
    Create f child nodes of RN,
    for i=1 to f do
        Set $RN_i$ to $D_i$, where $D_i$ € RN
        $D_i = f_i$
        BuildTree($RN_i$)
    endfor
elseif
end
```

5 Experimental results

The experiments have been carried out on intel core i5, 32GB RAM, and Windows 10. Pandas, NumPy, nltk, sklearn, matplotlib packages, spacy, NMSLIB were used for loading the data pre-processing and results in the evaluation. The most popular PROMISE and PROMISE_exp software requirement datasets are not suitable for our research.

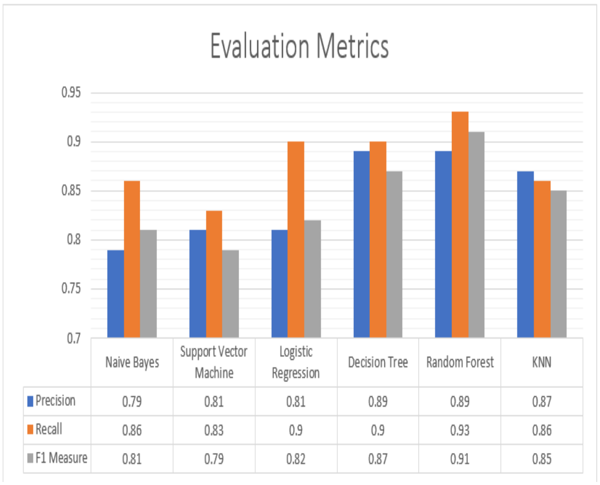
It is small in size, consisting of only 625 requirement instances, and the class distribution is also imbalanced. A novel dataset has been created regarding banking applications comprising 2812 requirement instances focusing on IReq, OReq, TFReq, and TSReq categories. The sample of requirement instances is illustrated in Figure 3 and 6 correspondingly. The prepared dataset is pre-processed, features extracted, requirements discovered, and classified using a python programming language. Spacy, a free, open-source library for NLP and NMSLIB, an efficient similarity search library, and a toolkit for evaluating search methods, which is the first principled support for non-metric space searching, is a significant part of programming. The performance of the Random Forest algorithm is compared with other supervised machine learning algorithms like Naïve Bayes, Support Vector Machine, Logistic Regression, KNN, etc. The evaluation metrics like Precision, Recall and F1 scores of 0.89, 0.93, and 0.91 respectively proves that RF is the best classification algorithm.

Evaluation metrics are primarily used to evaluate the performance of a classifier by comparing the predictions obtained by a model with the actual values in the corpus. The essential components for the metrics are True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). According to Hitesh et al. [45],

Precision = TP/TP+FP

Recall = TP/TP+FN

F1 Score = 2(Recall Precision) / (Recall + Precision)



	Naive Bayes	Support Vector Machine	Logistic Regression	Decision Tree	Random Forest	KNN
Precision	0.79	0.81	0.81	0.89	0.89	0.87
Recall	0.86	0.83	0.9	0.9	0.93	0.86
F1 Measure	0.81	0.79	0.82	0.87	0.91	0.85

Figure 13: Comparison of various algorithms with respect to metrics

Figure 13 shows the comparison of various supervised algorithms, out of which Random Forest records a higher value of precision with 0.89, recall of 0.93 and f1 measure of 0.91. Table 5 shows the performance results of each retrieved interactive type requirement.

Table 5: Results of random forest classification using word2vec

Random Forest Classification using Word2Vec			
Requirement Type	**Precision**	**Recall**	**F1 measure**
IReq	0.91	0.96	0.95
OReq	0.9	0.94	0.92
TFReq	0.89	0.91	0.9
TSReq	0.86	0.9	0.88

6 Conclusion

Based on the research results, it can be concluded that the appropriate identification of interactive requirements is vital for the successful development of software-intensive projects. The paper's novelty is the retrieval of interactive requirements, especially for DIAs. The retrieval of pertinent data will provide meaningful insights into business intelligence problems. Vectorizing the requirements documents with word embedding's using spacy is done to explore the documents with semantic features. As a result, it retrieved the interactive requirements separately as IReq, OReq, TFReq, and TSReq using a fast similarity (k -NN) search and NMSLIB. Also, it measured the impact of the extracted documents by comparing the performance with metrics acquired from training the Random Forest classifier on word count features. The result of precision, recall, and F1 are 0.89, 0.93, and 0.91, respectively. Therefore, retrieval of interactive requirements like IReq, OReq, TFReq, and TSReq help the developers to document their projects more effectively by minimizing the rework. However, studies have shown that in an unbalanced data set, automatic classification performs worse when the size of requirements of some labels is smaller. As future work, we plan to increase the requirements dataset and look for ways to mitigate the unbalance of the base, being able to improve the classification with little training data.

References

[1] P. Wang, K. Tao, C. Gao, X. Ning, S. Gu, and B. Deng, “Eliciting big data requirement from big data itself: A task-directed approach,” 2017 6th International Workshop on Software Mining (SoftwareMining), Nov. 2017. [Online]. Available: 10.1109/softwaremining.2017.8100849.

[2] C. Palomares, C. Quer, and X. Franch, “Requirements reuse and requirement patterns: a state of the practice survey,” Empirical Software Engineering, vol. 22, no. 6, pp. 2719–2762, Dec. 2016. [Online]. Available: 10.1007/s10664-016-9485-x.

[3] W. N. Robinson, S. D. Pawlowski, and V. Volkov, “Requirements interaction management,” ACM Computing Surveys, vol. 35, no. 2, pp. 132–190, Jun. 2003. [Online]. Available: 10.1145/857076.857079.

[4] H. Meth, M. Brhel, and A. Maedche, “The state of the art in automated requirements elicitation,” Information and Software Technology, vol. 55, no. 10, pp. 1695–1709, Oct. 2013. [Online]. Available: 10.1016/j.infsof.2013.03.008.

[5] Pohl K. Requirement’s engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant. Rocky Nook, Inc.; 2016 Apr 30.

[6] C. Li, L. Huang, J. Ge, B. Luo, and V. Ng, “Automatically classifying user requests in crowdsourcing requirements engineering,” Journal of Systems and Software, vol. 138, pp. 108–123, Apr. 2018. [Online]. Available: 10.1016/j.jss.2017.12.028.

[7] N. H. Madhavji, A. Miranskyy, and K. Kontogiannis, “Big Picture of Big Data Software Engineering: With Example Research Challenges,” 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering, May 2015. [Online]. Available: 10.1109/bigdse.2015.10.

[8] E. Sodagari and M. Keyvanpour, “Challenges Classification of Software Requirements Interaction Management Using Search-Based Methods,” 2019 5th International Conference on Web Research (ICWR), Apr. 2019. [Online]. Available: 10.1109/icwr.2019.8765253.

[9] M. Binkhonain and L. Zhao, “A review of machine learning algorithms for identification and classification of non-functional requirements,” Expert Systems with Applications: X, vol. 1, p. 100001, Apr. 2019. [Online]. Available: 10.1016/j.eswax.2019.100001.

[10] R. R. R. Merugu and S. R. Chinnam, “Automated cloud service based quality requirement classification for software requirement specification,” Evolutionary Intelligence, vol. 14, no. 2, pp. 389–394, May 2019. [Online]. Available: 10.1007/s12065-019-00241-6.

[11] C. SenthilMurugan and S. Prakasam, “A Literal Review of Software Quality Assurance,” International Journal of Computer Applications, vol. 78, no. 8, pp. 25–30, Sep. 2013. [Online]. Available: 10.5120/13511-1279.

[12] W. A. Qader, M. M. Ameen, and B. I. Ahmed, “An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges,” 2019 International Engineering Conference (IEC), Jun. 2019. [Online]. Available: 10.1109/iec47844.2019.8950616.

[13] J. Slankas and L. Williams, “Automated extraction of non-functional requirements in available documentation,” 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), May 2013. [Online]. Available: 10.1109/naturalise.2013.6611715.

[14] R. Jindal, R. Malhotra, and A. Jain, “Automated classification of security requirements,” 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2016. [Online]. Available: 10.1109/icacci.2016.7732349.

[15] M. Lu and P. Liang, “Automatic Classification of Non-Functional Requirements from Augmented App User Reviews,” Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Jun. 2017. [Online]. Available: 10.1145/3084226.3084241.

[16] Z. Kurtanovic and W. Maalej, “Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning,” 2017 IEEE 25th International Requirements Engineering Conference (RE), Sep. 2017. [Online]. Available: 10.1109/re.2017.82.

[17] R. Deocadez, R. Harrison, and D. Rodriguez, “Automatically Classifying Requirements from App Stores: A Preliminary Study,” 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), Sep. 2017. [Online]. Available: 10.1109/rew.2017.58.

[18] H. Liang, X. Sun, Y. Sun, and Y. Gao, “Text feature extraction based on deep learning: a review,” EURASIP Journal on Wireless Communications and Networking, vol. 2017, no. 1, Dec. 2017. [Online]. Available: 10.1186/s13638-017-0993-1.

[19] R. Dzisevic and D. Sesok, “Text Classification using Different Feature Extraction Approaches,” 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Apr. 2019. [Online]. Available: 10.1109/estream.2019.8732167.

[20] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, “YAKE! Keyword extraction from single documents using multiple local features,” Information Sciences, vol. 509, pp. 257–289, Jan. 2020. [Online]. Available: 10.1016/j.ins.2019.09.013.

[21] W. A. Qader, M. M. Ameen, and B. I. Ahmed, “An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges,” 2019 International Engineering Conference (IEC), Jun. 2019. [Online]. Available: 10.1109/iec47844.2019.8950616.

[22] M. Lu and P. Liang, “Automatic Classification of Non-Functional Requirements from Augmented App User Reviews,” Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Jun. 2017. [Online]. Available: 10.1145/3084226.3084241.

[23] S. Qaiser and R. Ali, “Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents,” International Journal of Computer Applications, vol. 181, no. 1, pp. 25–29, Jul. 2018. [Online]. Available: 10.5120/ijca2018917395.

[24] M. Bounabi, K. El Moutaouakil, and K. Satori, “Text classification using Fuzzy TF-IDF and Machine Learning Models,” Proceedings of the 4th International Conference on Big Data and Internet of Things, Oct. 2019. [Online]. Available: 10.1145/3372938.3372956.

[25] R. Lakshmi and S. Baskar, “Novel term weighting schemes for document representation based on ranking of terms and Fuzzy logic with semantic relationship of terms,” Expert Systems with Applications, vol. 137, pp. 493–503, Dec. 2019. [Online]. Available: 10.1016/j.eswa.2019.07.022.

[26] T. Walkowiak, S. Datko, and H. Maciejewski, “Bag-of-Words, Bag-of-Topics and Word-to-Vec Based Subject Classification of Text Documents in Polish - A Comparative Study,” Advances in Intelligent Systems and Computing, pp. 526–535, May 2018. [Online]. Available: 10.1007/978-3-319-91446-6_49.

[27] A. I. Kadhim, “Term Weighting for Feature Extraction on Twitter: A Comparison Between BM25 and TF-IDF,” 2019 International Conference on Advanced Science and Engineering (ICOASE), Apr. 2019. [Online]. Available: 10.1109/icoase.2019.8723825.

[28] K. S. Kalaivani, S. Uma, and C. S. Kanimozhiselvi, “A Review on Feature Extraction Techniques for Sentiment Classification,” 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Mar. 2020. [Online]. Available: 10.1109/iccmc48092.2020.iccmc-000126.

[29] E. Dias Canedo and B. Cordeiro Mendes, “Software Requirements Classification Using Machine Learning Algorithms,” Entropy, vol. 22, no. 9, p. 1057, Sep. 2020. [Online]. Available: 10.3390/e22091057.

[30] Md. A. Haque, Md. Abdur Rahman, and M. S. Siddik, “Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study,” 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), May 2019. [Online]. Available: 10.1109/icasert.2019.8934499.

[31] M. Lima, V. Valle, E. Costa, F. Lira, and B. Gadelha, “Software Engineering Repositories,” Proceedings of the XXXIII Brazilian Symposium on Software Engineering, Sep. 2019. [Online]. Available: 10.1145/3350768.3350776.

[32] R. Deocadez, R. Harrison, and D. Rodriguez, “Automatically Classifying Requirements from App Stores: A Preliminary Study,” 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), Sep. 2017. [Online]. Available: 10.1109/rew.2017.58.

[33] B. Behera and G. Kumaravelan, “Text document classification using fuzzy rough set based on robust nearest neighbor (FRS-RNN),” Soft Computing, vol. 25, no. 15, pp. 9915–9923, Nov. 2020. [Online]. Available: 10.1007/s00500-020-05410-9.

[34] A. Sainani, P. R. Anish, V. Joshi, and S. Ghaisas, “Extracting and Classifying Requirements from Software Engineering Contracts,” 2020 IEEE 28th International Requirements Engineering Conference (RE), Aug. 2020. [Online]. Available: 10.1109/re48521.2020.00026.

[35] I. M. S. Raharja and D. O. Siahaan, “Classification of Non-Functional Requirements Using Fuzzy Similarity KNN Based on ISO / IEC 25010,” 2019 12th International Conference on Information &amp; Communication Technology and System (ICTS), Jul. 2019. [Online]. Available: 10.1109/icts.2019.8850944.

[36] R. Deocadez, R. Harrison, and D. Rodriguez, “Automatically Classifying Requirements from App Stores: A Preliminary Study,” 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), Sep. 2017. [Online]. Available: 10.1109/rew.2017.58.

[37] R. Jindal, R. Malhotra, and A. Jain, “Automated classification of security requirements,” 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep. 2016. [Online]. Available: 10.1109/icacci.2016.7732349.

[38] H. Alrumaih, A. Mirza, and H. Alsalamah, “Domain Ontology for Requirements Classification in Requirements Engineering Context,” IEEE Access, vol. 8, pp. 89899–89908, 2020. [Online]. Available: 10.1109/access.2020.2993838.

[39] S. L. Lim and A. Finkelstein, “Anticipating Change in Requirements Engineering,” Relating Software Requirements and Architectures, pp. 17–34, 2011. [Online]. Available: 10.1007/978-3-642-21001-3_3.

[40] D. Virmani and S. Taneja, “A Text Preprocessing Approach for Efficacious Information Retrieval,” Advances in Intelligent Systems and Computing, pp. 13–22, Jun. 2018. [Online]. Available: 10.1007/978-981-10-8968-8_2.

[41] D. Sarkar, “Text Analytics with Python,” 2016. [Online]. Available: 10.1007/978-1-4842-2388-8.

[42] D. Sarkar, “Natural Language Processing Basics,” Text Analytics with Python, pp. 1–68, 2019. [Online]. Available: 10.1007/978-1-4842-4354-1_1.

[43] M. Bokan, “Negative-Sampling Word-Embedding Method,” Scientific Journal of Astana IT University, vol. 10, pp. 15–21, Jun. 2022. [Online]. Available: 10.37943/elgd6408.

[44] L. Boytsov and B. Naidan, “Engineering Efficient and Effective Non-metric Space Library,” Lecture Notes in Computer Science, pp. 280–293, 2013. [Online]. Available: 10.1007/978-3-642-41062-8_28.

[45] M. Hitesh, V. Vaibhav, Y. J. A. Kalki, S. H. Kamtam, and S. Kumari, “Real-Time Sentiment Analysis of 2019 Election Tweets using Word2vec and Random Forest Model,” 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), Sep. 2019. [Online]. Available: 10.1109/icct46177.2019.8969049.

[46] J. M. Perez-Verdejo, A. J. Sanchez-Garcia, and J. O. Ocharan-Hernandez, “A Systematic Literature Review on Machine Learning for Automated Requirements Classification,” 2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT), Nov. 2020. [Online]. Available: 10.1109/conisoft50191.2020.00014.

[47] M. Kashina, I. D. Lenivtceva, and G. D. Kopanitsa, “Preprocessing of unstructured medical data: the impact of each preprocessing stage on classification,” Procedia Computer Science, vol. 178, pp. 284–290, 2020. [Online]. Available: 10.1016/j.procs.2020.11.030.

[48] A. K. Uysal and S. Gunal, “The impact of preprocessing on text classification,” Information Processing &amp; Management, vol. 50, no. 1, pp. 104–112, Jan. 2014. [Online]. Available: 10.1016/j.ipm.2013.08.006.

[49] M. Anandarajan, C. Hill, and T. Nolan, “Planning for Text Analytics,” Advances in Analytics and Data Science, pp. 27–41, Oct. 2018. [Online]. Available: 10.1007/978-3-319-95663-3_3.