Clustering-based Automated Requirement Trace Retrieval

Nejood Hashim Al-walidi[1], Shahira Shaaban Azab[2], Abdelaziz Khamis[3], Nagy Ramadan Darwish[4]
Department of Information Systems and Technology[1, 3, 4]
Department of Computer Science[2]
Faculty of Graduate Studies for Statistical Research, Cairo University, Cairo, Egypt

***Abstract*—The benefits of requirement traceability are well known and documented. The traceability links between requirements and code are fundamental in supporting different activities in the software development process, including change management and software maintenance. These links can be obtained using manual or automatic means. Manual trace retrieval is a time-consuming task. Automatic trace retrieval can be performed via various tools such as Information retrieval or machine learning techniques. Meanwhile, a big concern associated with automated trace retrieval is the low precision problem primarily caused by the term mismatches across documents to be traced. This study proposes an approach that addresses the term mismatch problem to obtain the greatest improvements in the trace retrieval accuracy. The proposed approach uses clustering in the automated trace retrieval process and performs an experimental evaluation against previous benchmarks. The results show that the proposed approach improves the trace retrieval precision.**

***Keywords*—Requirements traceability; information retrieval; term mismatch problem; trace retrieval; TraceLab; clustering**

I. Introduction

Many software development standards have been proposed in response to the high rate of software project failures. These standards include SEI's CMMI and IEEE's JSTD-016. A common feature of these standards is that they all impose requirement traceability (RT) practices on the software development process [1]. RT is an important quality factor of software development that intends to ensure a continued alignment between stakeholder requirements and various outputs of the system development process. In addition [2], RT makes it easy to determine what software artifacts must be updated to fulfill a change request made during the maintenance phase of the software project.

In 2014, members of the Center of Excellence for Software traceability (CoEST) established a roadmap for advancing the state of practice in software traceability and presented a focused research agenda for software traceability. They identified seven broad research focus areas and outlined the specific research needed in each of these areas. Under the umbrella of one of these research areas, that is, *Creating & Maintaining Traces*, we focus herein on trace retrieval, which is concerned with dynamically generating trace links between source and target artifacts [3]. Researchers utilized different algorithms to infer the trace links between related artifacts based on the assumption that related artifacts contain related terms [4]. The underlying concept behind this was that these algorithms could be used to estimate the similarity between two documents.

A big concern associated with trace retrieval research is the low precision problem. Precision is the percentage of correct traces over all retrieved traces. "Low precision" indicates that many false traces have been incorrectly retrieved, and the user must manually evaluate the retrieved links to identify the correct traces. This leads to an emphasis on precision. Accordingly, some researchers examined ideas on how to increase it [4] [5]; however, these ideas resulted in only a minor increase in precision. The problem is primarily caused by term mismatches across documents to be traced. The reason for choosing the proposed approach is to enhance trace retrieval precision. Therefore, the authors formulated the problem by choosing an intelligent solution based on unsupervised learning using clustering. The datasets have a limited number of labels. These labels are provided for testing only. So, the suggestion for addressing this problem is by choosing an intelligent solution based on unsupervised learning to find similarities in the data point and group similar data points together to enhance trace retrieval precision.

In this work, to address the term mismatch problem in automated trace retrieval, we follow the proposed research direction toward achieving automated trace retrieval, that is, to develop intelligent tracing solutions, "which are not constrained by the terms in source and target artifacts, but which understand domain-specific concepts, and can reason intelligently about relationships between artifacts" [3]. We lay down the foundations needed to use clustering in automated trace retrieval between *source code* and *requirements*.

Clustering, which relies on unsupervised machine learning, is the task of grouping a set of objects in such a way that objects in the same group, called a cluster, are more similar in some sense to each other than to those in other groups or clusters [6]. In this study, we used K-means++, Hierarchical, Gaussian mixture model (GMM), and Density-based Spatial Clustering of Application with Noise (DBSCAN) algorithms. Clustering algorithms can be summarized in three steps:

1) *Inputs*: dataset after preprocessing and other steps related to the dataset type.
2) *Applying* suitable models.

A. Outputs: evaluating the results after the clustering processes; for this, we decided to use clustering to cluster the dataset into two: clusters 0 and 1. Cluster 1 means the

items are similar. Cluster 0 means the items are different. All clustering steps are described in Section IV. We present an overview of these methods below.

The K-means is a partition-based iterative clustering algorithm, in which each cluster is characterized by its center point. It is widely used because of its implementation simplicity and effectiveness. It assumes that the number of clusters represented by "K" in K-means is already known. This algorithm aims to minimize the distance inside the same cluster and maximize the distance between the clusters [7].

Hierarchical clustering divides the data set at different levels to form a tree-shaped cluster structure that naturally defines clusters by branches in the hierarchical tree. It makes a few assumptions regarding the overall data point distribution; hence, it is suitable for datasets of many different shapes. Hierarchical clustering has two main implementations: agglomerative and divisive clustering [8].

In model-based clustering, each cluster is considered as a generative model with mean and variance. Instances arise from a distribution that is a mixture of several components. Gaussian (normal) distributions are the most used representation in model-based clustering. The mixture model is the GMM, whose components are Gaussian distributions with different means and variances [9].

The density-based clustering algorithm, called DBSCAN, recognizes arbitrary-shaped clusters under a high noise level of the studied data. The efficiency of DBSCAN depends on the parameter values set up at the initial step of the algorithm usage [10].

In this paper, we adopted four widely used metrics to evaluate the new proposed approach, precision, recall, F-score, and accuracy. Both precision and recall are used to assess the effectiveness of the requirement tracing tools [11]. **Precision** is defined as the percentage of correct retrieved candidate links and measures the fraction of retrieved documents which are relevant [12]. **Recall** is the percentage of the found correct links [13]. **F-score** is used to provide a balance between recall and precision [14]. It is the harmonic mean of recall and precision [15] and used to aggregate both measures into a single value [16]. Finally, **accuracy** measures are the rate of normal and outlier values correctly classified between the total number of classifications [17].

The remainder of this paper is structured as follows: Section II describes the related works in the trace retrieval field; Section III presents the proposed approach; Section IV explains the obtained results; Section V includes a result evaluation; and Section VI provides the conclusions and future work.

II. Related Works

This section reviews some methods used to solve the term mismatch problem in automated requirement trace retrieval.

In the paper [19] titled "Automatic traceability link recovery via active learning", the authors proposed a new traceability link recovery approach based on active learning (AL). They evaluated their approach on seven datasets used in traceability and compared them with an information retrieval (IR)-based approach and a state-of-the-art machine learning approach, called traditional supervised learning. The used datasets are available from the CoEST website (http://www.CoEST.org). The results showed that the AL-based approach outperforms the other two in terms of the F-score.

In the paper [20] titled "Traceability Transformed: Generating more Accurate Links with Pre-Trained BERT Models", the authors proposed a framework, called Trace BERT (T-BERT), to create trace links between the source code and the natural language artifacts. They then applied the T-BERT framework to recover links between issues and commits in open-source projects. The evaluation results as regards the accuracy and efficiency of three BERT architectures indicated that the Single-BERT architecture generated the most accurate links, while the Siamese-BERT architecture produced comparable results with significantly less execution time. By learning and transferring knowledge, all three models in the framework outperformed classical IR trace models.

In the paper [21] titled "Analyzing close relations between target artifacts for improving IR-based requirement traceability recovery", the authors proposed a method for trace link recovery by combining the IR method and the close relations between the target artifacts. This approach was referred to as IR_CRT. Aside from textual similarity, the close semantic relations between the target artifacts were considered. Experiments on five public datasets indicated that the precision on these datasets improved by 15.6% on average, showing that their method outperformed the baseline when working under the same conditions.

In the paper [22] titled "Ontology-based Trace Retrieval", the authors solved the term mismatch problem in automated requirement trace retrieval by incorporating information from the general and domain-specific ontologies into the tracing process. They used ontologies to identify relationships that would not be recognized by standard IR techniques. They then experimentally evaluated their approach against the standard vector space model (VSM). Their results showed that a domain ontology combined with a generalized ontology returns the greatest improvements in trace accuracy.

In the paper [23] titled "Towards an Intelligent Domain-Specific Traceability Solution", the authors solved the term mismatch problem in automated requirement trace retrieval by presenting the domain-contextualized intelligent traceability (DoCIT) solution. This approach mimicked some of the higher-level reasoning that a human trace analyst performs. The authors focused their efforts on the complex domain of communication and control in a transportation system and found that their approach can significantly improve the quality of the generated trace links. They illustrated and evaluated DoCIT with examples and experiments from the control and communication sector of a transportation domain.

In the paper [24] titled "Combining Machine Learning and Logical Reasoning to Improve Requirements Traceability Recovery", the authors proposed a novel traceability link recovery approach that measures the similarity between requirements and the source code by exploring their features.

They combined machine learning and logical reasoning models and conducted a series of experiments on four datasets to evaluate the performance of their method against existing approaches. Their experiments showed that their approach is substantially better than other methods.

III. Proposed Approach

In this section, we divided the task of the proposed approach using clustering to improve the recovery of links between source code and requirements, into two main phases. Fig. 1 illustrates the two phases of the proposed trace retrieval approach.

B. Phase 1: Choosing Suitable Datasets

Research on automated requirement traceability relies on the availability of different dataset types. In general, obtaining datasets has been one of the reported barriers for researchers in the software engineering domain [25]. This phase introduces the three datasets used in this study to evaluate automated trace retrieval between the source code and requirements. Table I defines the characteristics of the datasets used in the proposed approach. These datasets are available from http://www.CoEST.org.

TABLE I. Dataset Characteristics [25],[26],[27], [28][29][30]

Dataset description	Freq	Traceability details	Trace space	CC	UC	Trace links
eTour: "an electronic tourist guide developed by students"	10	Use cases to code	6728	116	58	366
SMOS: "an application that is used to monitor high school students (e.g., absence, grades)"	7	Use cases to classes	6700	100	67	1044
eANCI: "system providing support to manage Italian municipalities"	3	Use cases to classes	7645	55	140	567

The above-mentioned datasets were chosen because they are available for all researchers and compatible with TraceLab's environment.

With these datasets, we ran into some issues related to the two artifacts (requirements and source code). For example, all three datasets are imbalanced datasets. They included useless words, with some words starting with extra letters and ending with extra letters without meaning. A few single characters can be found between the lines of the files that did not have a clear meaning. According to the Answer set of the dataset, some source codes did not have any requirements, which affected the links between the two artifacts and gave wrong similarities. Some words and sentences were written in different languages. Some of them were in English, while others were in Italian.

C. Phase 2: Choosing and Applying Clustering Models

Clustering is an unsupervised machine learning task used to gather data into many collections or clusters according to the similarities of the data point features and characteristics [31]. In this phase, we followed seven main steps to apply clustering models to the three datasets:

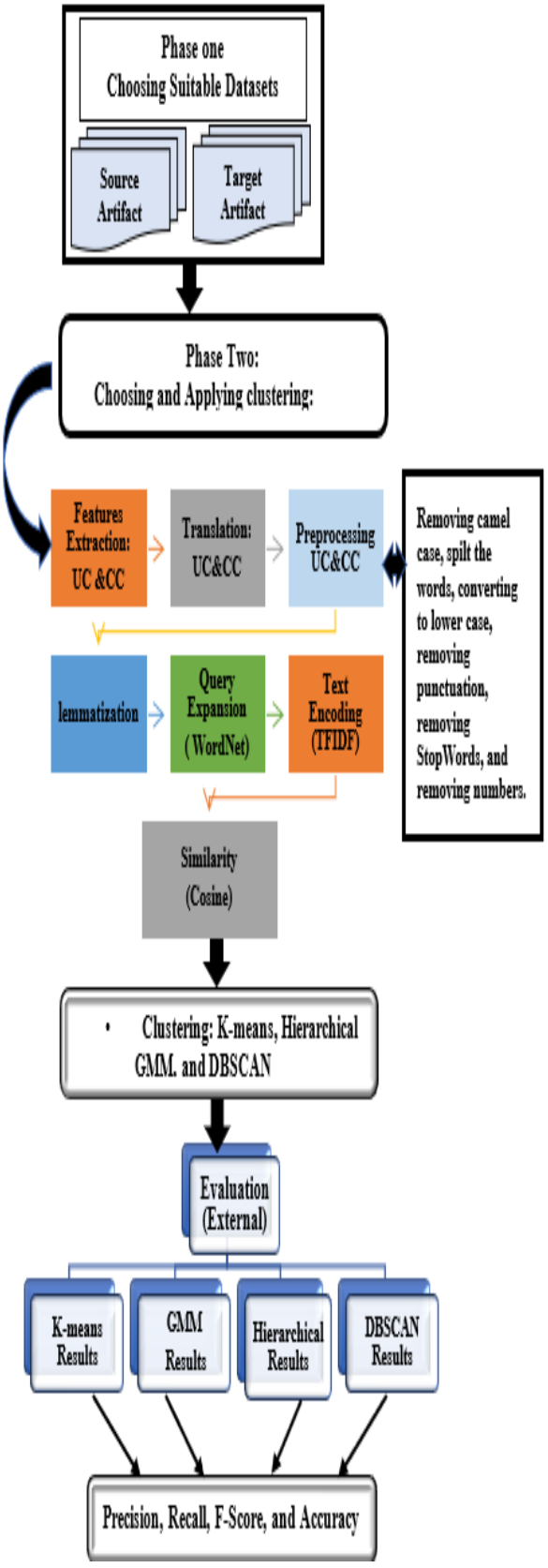


Fig. 1. Proposed trace retrieval approach

Step 1, *Feature extraction*: The requirements (source artifact) and the source code (target artifact) were written in natural and programming languages, respectively. Hence, they included different types of information that were not all useful for identifying the traceability links between the two artifacts. For that, we extracted the most important source code feature, namely (*class_name, class_attribute, class_comments, method_comment,method_name,method_parameter,method_return*). We also extracted the two main features related to the requirements (i.e., Title and Description) as shown in Fig. 3.

Step 2, *Translation*: The content of the two artifacts (i.e., UC and CC files) included Italian words and sentences in some files and some parts of the file lines. Therefore, the content of the two artifacts was translated into English. To this end, we leveraged the advanced translation engine to translate documents into English if they were originally unspecified in English. We used one of the tools used for translation into the English language namely Google Translate [32], which enables a method of translating text from one language to another.

Step 3, *Preprocessing*: We followed a classic process to pre-process the contents of the software artifacts, UC and CC. The preprocessing comprised the following steps: removing the camel case, splitting the words, converting to a lower case, and removing punctuations, Stop Words, and numbers. We performed lemmatization for all words to facilitate natural language analysis, such as transforming the verb, noun, adjectives, and adverbs into their bases and examining the study of word texts with the intention of finding something that adds meaning to the word text [33] and removing any single character from both artifacts.

Step 4, *Query expansion using WordNet*: WordNet is an online lexical system (database) for the English language that provides diverse and wide-ranging semantic information [34]. We used query expansion to add related words to a query to increase the number of returned documents and improve the recall accordingly. In most cases, all words in each query should first be extracted. For each word, the synonyms were automatically selected.

Step 5, *Text encoding*: Many methods can be used to convert text into numerical vectors, such as TF-IDF encoding, Dec2Vec, Word2Vec, and Bag of Words (BOW) [35]. We experimented Dec2Vec and Word2Vec, but they did not improve experiments results, so we used the TFIDF to have better results.

Step 6, *Computing similarity*: Using the cosine similarity [36], we computed the similarity between two vectors after converting the corresponding contents of each feature into a vector by using the TFIDF algorithm. The cosine similarity algorithm measures the similarity between two vectors, which can represent paragraphs, sentences, words, or the entire document.

Step 7: In this step, four types of clustering models were used, namely K-means++, GMM (hard clustering type), Hierarchical, and DBSCAN. After the previous steps, we first prepared the dataset. Fig. 2 illustrates the total columns and rows of the dataset which are 14 columns. Second, we chose the clustering type. Third, evaluating the results, we performed external evaluation criteria and computed the confusion matrix (i.e., precision, recall, F-score, and accuracy measures. Table II shows the meaning of the confusion matrix symbols that are used for precision, recall, F-Score, and accuracy. The mathematical formulas of accuracy, precision, recall, and F-score are as follows:

Accuracy = $(TP + TN) / (TP + FP + FN + TN)$.

Precision = $TP / (TP + FP)$

Recall = $TP / (TP + FN)$

F-score = 2*(precision*recall)/ (precision + recall).

TABLE II. MEANING OF THE CONFUSION MATRIX SYMBOLS [18]

Symbol	Description
TP	The number of data pairs found in the same cluster, both in C and in P.
FP	The number of data pairs found in the same cluster in C but in different clusters in P.
FN	The number of data pairs found in different clusters in C but in the same cluster in P.
TN	The number of data pairs found in different clusters, both in C and in P.

C means the actual cluster and P means the prediction cluster.

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
0	0	0	0	0	0	0	0	0	0.1026	0.1003	0	0.0231	0
0	0	0	0	0	0	0	0	0.0277	0.0795	0.0242	0	0.0606	0.0247
0	0	0	0	0	0	0	0	0	0.0216	0.0135	0.0193	0.0273	0.0214
0	0	0	0	0	0	0	0	0	0	0.0305	0	0	0.0172
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.1127	0	0.0115	0	0	0	0	0.1301	0	0.0095	0	0
0	0	0	0	0	0	0	0	0.0284	0	0	0.0371	0	0.0279
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.3077	0	0.2007	0	0.0672	0	0
0	0	0	0	0	0	0	0.1521	0.0122	0	0	0.0286	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.0036	0	0
0	0.0795	0.1109	0	0.057	0.0705	0.0255	0	0.0787	0.0916	0	0.0471	0.0814	0.0211
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.0247	0.0835	0.189	0.044	0.1305	0.0109	0	0.0519	0.2206	0.1859	0.0508	0.1178	0.0126
0.1465	0.1176	0.0473	0.0685	0.1205	0.1713	0	0.2479	0.1165	0.0583	0.0954	0.0995	0.1554	0.037
0	0.0172	0	0	0.0115	0	0	0.0953	0.0142	0.0485	0.0255	0.0202	0.0089	0
0	0	0	0	0	0	0	0.0716	0.0634	0	0.0128	0	0	0
0.1279	0.0774	0.0507	0.0503	0.0774	0.1032	0	0.2165	0.0894	0.0419	0.0415	0.0894	0.0852	0.0288
0	0	0	0	0	0	0	0	0.0276	0.1323	0.0322	0.028	0	0
0	0	0	0	0	0	0	0	0.0138	0.0354	0	0.0187	0	0

Fig. 2. The 14 columns of the dataset

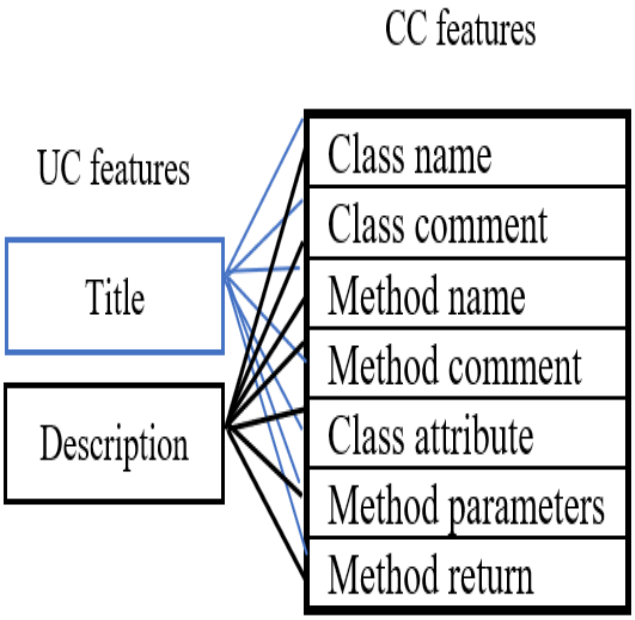


Fig. 3. Feature extraction

IV. RESULTS

In this section, we present the results of applying the Proposed Trace Retrieval Approach described in Section 3, which consists of two main phases as shown in Fig 1.

As mentioned in Section III, we applied four clustering models (i.e., K-means++, GMM, Hierarchical, and DBSCAN) to the three datasets described in Phase 1 of the proposed trace retrieval approach.

The results of the new proposed trace retrieval approach are presented here which include four measures: Precision, recall, F-score, and accuracy as shown in (Table III).

Comparing the proposed approach with two experiments, namely TraceLab and IR as shown in Tables IV (A) to (C). Tables V (A) to (C) show the comparison results of the proposed method and the two studies (i.e., studies [19] and [24]).

First, Table III and Fig. 4 present the results of K-means++ for three datasets: eTour, SMOS, and eANCI achieved high values compared to GMM, hierarchical DBSCAN results. The proposed trace retrieval approach is effective in improving precision, recall, F-score, and accuracy. The extent of improvement differed from one dataset to another depending on the dataset type. We used accuracy as an additional measure, whereas TraceLab and Information retrieval R and two studies [19] and [24] did not. The proposed approach achieved highest results using accuracy with the three datasets (i.e., eTour = 0.91, SMOS = 0.66, and eANSI = 0.60). Three parameters used in the experiments: max_feature with a value of 400 for the eTour dataset, SMOS, featurewiz when (corr_limit values) equal 0.5, and corr_limit values equal 0.5 for eANSI.

TABLE III. SUMMARY OF THE RESULTS OF THE PROPOSED TRACE RETRIEVAL APPROACH USING TWO CLUSTERING ALGORITHMS FOR THREE DATASETS

eTour dataset:				
Model Name	**Precision**	**Recall**	**FScore**	**Accuracy**
K-means++	**0.93**	**0.97**	**0.94**	**0.91**
SMOS dataset				
K-means++	**0.73**	**0.76**	**0.74**	**0.66**
eANCI dataset				
K-means++	**0.64**	**0.77**	**0.70**	**0.60**

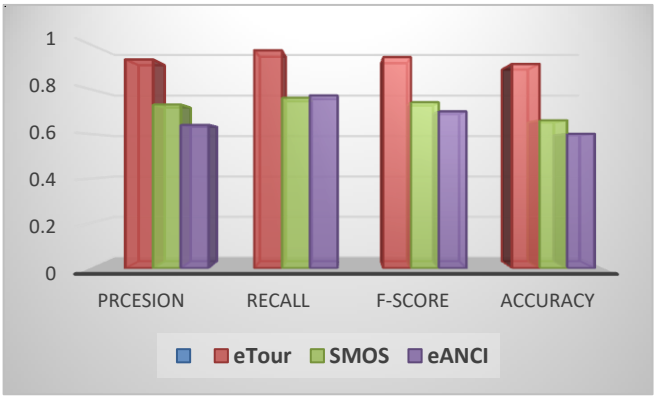
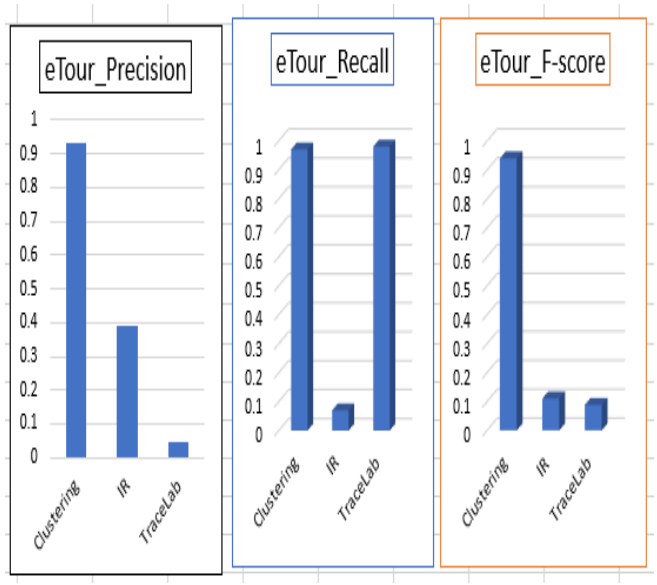
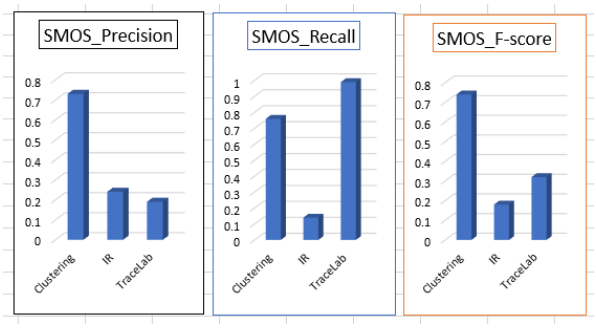


Fig. 4. Summary of the results of the proposed trace retrieval approach using two clustering algorithms for three datasets

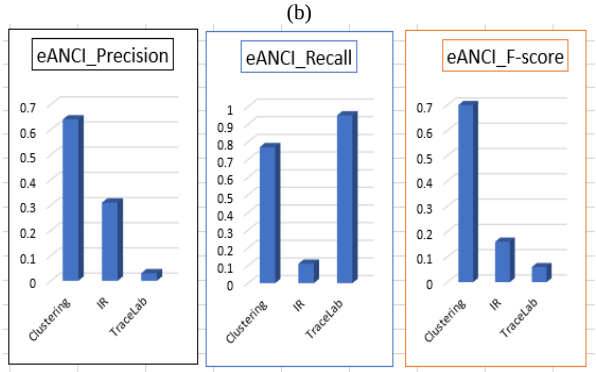
Second, presenting a comparison between the proposed trace retrieval approach and two experiments, namely ***TraceLab and IR***. Table IV (A) to (C) and Fig. 5(a) to (c) present a comparison of the proposed approach and the two experiments. The results showed that the former returns the greatest improvements in the trace measures.



(a)



(b)



(c)

Fig. 5. (a). eTour result comparison: clustering, IR, and TraceLab (b) SMOS result comparison: clustering, IR, and TraceLab (c) eANCI result comparison: clustering, IR, and TraceLab

1) Experiment one (TraceLab): Study [37] designed to help and support the reproducibility of experiments in software engineering and maintenance. In TraceLab's visual modeling environment [38], IR algorithms are implemented as experiments using a library of reusable and user-defined components [39]. An already existing experiment, called Basic IR [40], which is a standard support vector model experiment, was conducted to implement the VSM/TF*IDF algorithm.

2) Experiment two: four steps were applied. *The first step* was preprocessing, which included transforming the upper case to the lower case, removing the punctuations and digits, converting plurals to singulars, and transforming verbs into infinitives. *The second* step was getting the weight of each term into artifacts using TFIDF. *The third step* was computing the similarity between the source and target artifacts using the cosine similarity. *The fourth* and final step was computing the precision, recall, and F-score.

TABLE IV. (A). ETOUR COMPARED WITH IR AND TRACELAB

Measure	Clustering	IR	TraceLab
Precision	**0.93**	0.39	0.046

Measure	Clustering	IR	TraceLab
Recall	**0.97**	0.07	0.98

Measure	Clustering	IR	TraceLab
F-score	**0.94**	0.11	0.088

(B). SMOS COMPARED WITH IR AND TRACELAB

Measure	Clustering	IR	TraceLab
Precision	**0.73**	0.24	0.19

Measure	Clustering	IR	TraceLab
Recall	0.76	0.14	0.99

Measure	Clustering	IR	TraceLab
F-score	**0.74**	0.18	0.32

(C). EANCI COMPARED WITH IR AND TRACELAB

Measure	Clustering	IR	TraceLab
Precision	**0.64**	0.31	0.03

Measure	Clustering	IR	TraceLab
Recall	**0.77**	0.11	0.95

Measure	Clustering	IR	TraceLab
F-score	**0.70**	0.16	0.06

Tables IV (A) to (C) and Fig. 5(a) to (c) show that the proposed trace retrieval approach achieved high results with three datasets in the precision measure, which was the concern in this work. TraceLab achieved high results in recall because the precision and F-score measures were very low, and the false negative (FN) of the links was more than the true positive with the three datasets. The proposed approach achieved high results with the three datasets in F-score compared to the IR and TraceLab.

Third, compared the results of the proposed approach and those of other studies. Tables V (A) to (C) and Fig. 6(a) to (c) present the results obtained from applying clustering models into the three datasets. The performance of the proposed approach was compared to those of studies 19 and 24, which addressed the same problem and used the same datasets. Four measures were used: precision, recall, and F-score as the comparison metrics. The two studies used precision, recall, and F-score, but not accuracy.

Tables V (A) to (C) and Fig. 6(a) to (c) depict that the proposed trace retrieval approach achieved high results with precision, recall, and F-Score in the eTour dataset than in the two studies. In the SMOS dataset, the proposed approach trace retrieval achieved high results using precision, recall, F-Score than in the two studies. Meanwhile, in the eANCI dataset, the proposed approach obtained low results using precision than in the two studies and highest results in recall and F-Score.

TABLE V. (A). ETOUR COMPARED WITH THE TWO STUDIES

Measure	Clustering	Study [19]	Study [24]
Precsion	**0.93**	0.68	0.66

Measure	Clustering	Study [19]	Study [24]
Recall	**0.97**	0.34	0.59

Measure	Clustering	Study [19]	Study [24]
F-score	**0.94**	0.46	0.61

(B). SMOS COMPARED WITH THE TWO STUDIES

Measure	Clustering	Study [19]	Study [24]
Precsion	**0.73**	0.57	0.75

Measure	Clustering	Study [19]	Study [24]
Recall	**0.76**	0.29	0.33

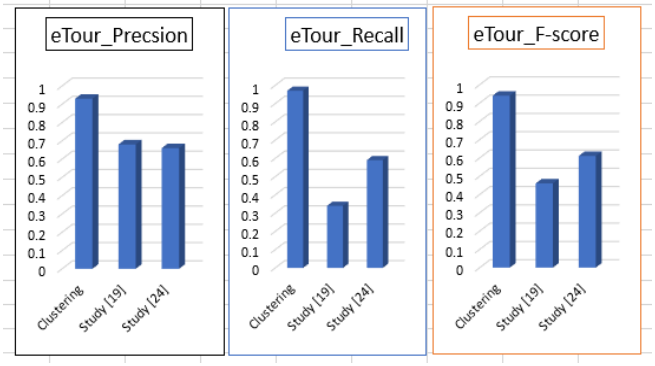
Measure	Clustering	Study [19]	Study [24]
F-score	**0.74**	0.39	0.51

(C). EANCI COMPARED WITH THE TWO STUDIES

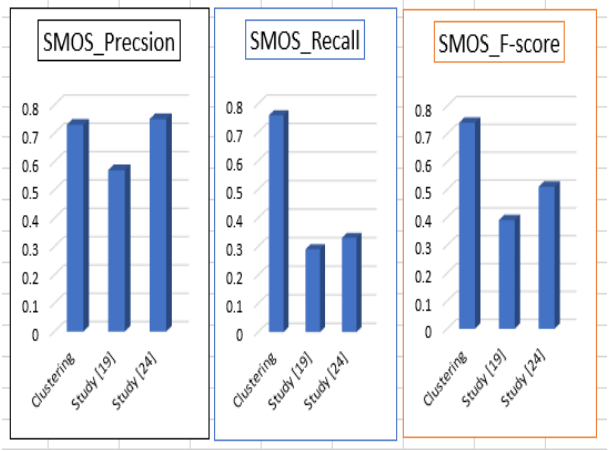
Measure	Clustering	Study [19]	Study [24]
Precsion	**0.64**	0.73	0.62

Measure	Clustering	Study [19]	Study [24]
Recall	**0.77**	0.44	0.54

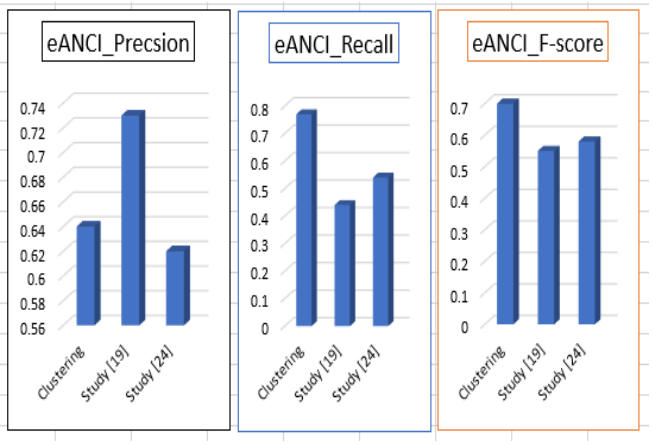
Measure	Clustering	Study [19]	Study [24]
F-score	0.70	0.55	0.58



(a)



(b)



(c)

Fig. 6. (A). eTour result comparison: clustering and studies [19] and [24] (B). SMOS result comparison: clustering and studies [19] and [24] (C). eANCI result comparison: clustering and studies [19] and [24]

Finally, the authors will present all the experimental results obtained from K-means++, hierarchical, and DBSCAN as shown in (Tables VI-XVII). Those tables include different results using various parameters used during running the experiments such as (max_features). The max_features set the maximum number of features to be used by specifying the value between (100 and 900). Also, featureswiz package was used. The package is available at https://pypi.org/project/featurewiz/. featurewiz package is used for selecting the most important features using different parameters like ***corr_limit*** parameter using different values (0.1 – 0.9) as follows:

features, train = featurewiz (data, target, ***corr_limit=0.9****, verbose=2, sep=",", header=0, test_data=" ", feature_engg=" ", category_encoders="").*

First: eTour dataset, (Tables VI and VII) presents the results of the experiments obtained from K-means++ and hierarchical.

Table VI shows the result of the proposed trace retrieval approach using K-means++ based on max_feature values (100,300,400,800). The Table indicated that the precision achieved the highest value with all max_feature values. Recall achieved the highest results when is max_feature values=400 and 800 and F-Score with max_feature values=300 and 400.Finally, the accuracy achieved highest results when is max_feature values=300,400, and 800.

TABLE VI. ETOUR_K-MEANS ++ (MAX_FEATURE)

Measure's name	max_feature values			
	800	400	300	100
Precision	0.93	**0.93**	0.93	0.93
Recall	0.97	0.96	0.96	0.95
F-Score	0.93	0.94	0.94	0.93
Accuracy	0.91	0.91	0.91	0.90

According to Table VII, the proposed trace retrieval approach using hierarchical based on the max_feature values achieved the highest results with precision, recall, F-Score, and accuracy when is max_feature value =950.

TABLE VII. ETOUR_ HIERARCHICAL (MAX_FEATURE)

Measure's name	Max_feature vlues			
	800	**400**	**300**	**950**
Precision	0.91	**0.91**	0.91	**0.93**
Recall	0.56	0.56	0.56	**0.95**
F-score	0.69	0.69	**0.69**	**0.93**
Accuracy	0.56	0.57	0.56	**0.90**

Second: SMOS dataset, (Tables VIII–XII) presents the results of the experiments obtained from K-means++ and hierarchical.

Table VIII indicates the results of the proposed trace retrieval approach using K-means++ based on max_feature values (100 - 500). Precision achieved the highest results when are max_feature values-=400. Both Recall and accuracy achieved the highest results when is max_feature values=500, and F-score achieved the highest results when are max_feature values=300

TABLE VIII. SMOS_ K-MEANS ++MAX_FEATURE (MINMAX_SCALARE)

Measure's name	max_feature values				
	500	**400**	**300**	**200**	**100**
Precision	**0.71**	**0.72**	0.71	0.71	**0.71**
Recall	**0.80**	**0.74**	**0.79**	**0.73**	0.69
F-score	0.75	0.73	**0.77**	**0.70**	0.68
Accuracy	**0.66**	0.65	0.65	0.64	0.62

The authors inferred that the corr_limit values enhance precision, recall, F-Score, and accuracy with different values as shown in Table IX. Precision achieves the highest results when the corr_limit values =0.7 and 0.5. Recall and accuracy obtain the highest results when the corr_limit values = 0.7. F-Score obtains the highest result when corr_limit values=0.7 and 0.5.

TABLE IX. SMOS_KMEANS++ USING FEATURE_WIZ (MIN, AXSCALER)

Measure's name	Corr_limit values			
	0.7	**0.5**	**0.4**	**0.3**
Precision	**0.72**	**0.72**	**0.71**	0.71
Recall	0.71	0.70	0.67	**0.67**
F-score	0.71	0.71	0.69	**0.69**
Accuracy	**0.64**	**0.63**	**0.61**	**0.61**

As shown in Table X, the proposed trace retrieval approach using K-means++ based on standard scalar (corr_limit values). Precision and accuracy achieve the highest results when corr_limit value=0.5. Also, the recall and F-Score achieved the highest results when is corr_limit value=0.3.

TABLE X. SMOS_KMEANS ++ USING (STANDARD SCALAR)

Measure's name	Standard scalar (corr_limit values)			
	0.5	**0.4**	**0.3**	**0.6**
Precision	**0.73**	**0.70**	**0.66**	0.71
Recall	**0.76**	0.75	**0.96**	0.77
F-score	**0.74**	0.72	**0.78**	0.74
Accuracy	**0.66**	**0.63**	0.65	0.64

Table XI lists the results of the proposed trace retrieval approach using hierarchical based on max_feature values (100,200, 300, and 500). The precision achieved the highest value with all values of the max_feature. Recall, F-Score, and accuracy obtains the highest value when is max_feature values =500.

TABLE XI. SMOS_ HIERARCHICAL MAX_FEATURE(MINMAX_SCALARE)

Measure's name	max_feature values			
	500	**300**	**200**	**100**
Precision	**0.71**	**0.71**	**0.71**	**0.71**
Recall	**0.80**	0.79	0.73	0.77
F-score	0.76	0.75	0.72	0.74
Accuracy	**0..66**	0.65	0.64	0.64

Table XII presents the proposed trace retrieval approach using hierarchical based on featurewiz using (corr_limit values). Precision achieved highest results when is corr_limit value=0.5. The recall, F-Score, and results enhance when corr_limit values equal 0.6. Accuracy when corr_limit equal 0.5 and 0.6

TABLE XII. SMOS _ HIERARCHICAL USING SELECT FEATURES (FEATUREWIZ)

Measure's name	Standard scalar (corr_limit values)			
	0.5	**0.6**	**0.4**	**0.8**
Precision	**0.73**	0.67	0.71	**0.71**
Recall	0.75	**0.94**	**0.68**	0.76
F-score	0.74	**0.78**	0.69	0.73
Accuracy	**0.66**	**0.66**	0.62	**0.65**

Third: eANCI dataset, (Tables XIII-XVII) presents the results of the experiments obtained from DBSCAN, K-means++, and hierarchical.

The proposed trace retrieval approach using DBSCAN achieved the highest value in precision when is corr_limit values-=0.5 and 0.7. Recall F-Score,and accuracy achieved the highest value when is corr_limit values=0.7 as shown in Table XIII.

TABLE XIII. EANCI_DBSCAN USING FEATUREWIZ

Measure's name	Standard scalar (corr_limit values		
	0.5	**0.6**	**0.7**
Precision	**0.64**	0.61	**0.64**
Recall	0.71	0.53	**0.77**
F-score	0.67	0.57	**0.70**
Accuracy	0.57	0.51	**0.60**

As shown in Table XIV, the proposed trace retrieval approach using K-means++ based on standard scalar achieves highest precision with all values of the Max_feature. Recall-Score, and accuracy achieves the highest results when are Max_feature =200.

TABLE XIV. EANCI_KMEANS ++ USING MAX_FEATURE

Measure's name	Max_feature		
	100	**200**	**300**
Precision	**0.63**	**0.63**	**0.63**
Recall	0.82	**0.84**	0.84
F-Score	**0.71**	**0.72**	**0.72**
Accuracy	0.59	0.60	**0.72**

From Table XV, the authors inferred that the proposed trace retrieval approach using K-means++ based on standard scalar (corr_limit values) achieves the highest results with precision when is corr_limit value=0.5 and 0.7. Recall, F-Score, and accuracy achieved the highest results when is the corr_limit value=0.7.

TABLE XV. EANCI_ KNEAMS USING FEATURERWIZ

Measure's name	Standard scalar (corr_limit values)		
	0.5	**0.6**	**0.7**
Precision	**0.64**	0.61	0.64
Recall	**0.71**	0.53	**0.77**
F-score	**0.67**	0.57	0.70
Accuracy	**0.57**	**0.51**	0.60

According to Table XVI, the proposed trace retrieval approach using hierarchical achieves the highest results for precision, Recall, F-Score, and accuracy when is MinMax scalar = 100.

TABLE XVI. EANCI_ HIERARCHICAL USING MAX_FEATURE

Measure's name	Standard scalar = 400	Standard scalar = 100
Precision	0.62	**0.63**
Recall	**0.68**	0.77
F-score	0.65	**0.69**
Accuracy	0.56	**0.59**

As shown in Table XVII, the proposed trace retrieval approach using a GMM based on a standard scalar achieves the highest value in precision, Recall, F-Score, and accuracy when is corr_limit values = 0.5.

TABLE XVII. EANCI _GMM_ USING FEATUREWIZ

Measure's name	Standard scalar (corr_limit values)	
	0.5	**0.6**
Precision	**0.63**	**0.61**
Recall	**0.65**	**0.52**
F-score	**0.63**	0.56
Accuracy	**0.56**	**0.50**

Regardless of the results, the suitable intelligent solution for addressing the term mismatch problem between requirements and source code for a few labels or unlabelled data is UNSUPERVISED machine learning using clustering, which is the only way to solve the problem of labels and enhance low precision.

V. EVALUATION

This section presents the evaluation method of the retrieved links evaluated with respect to two criteria. The first criterion is the performance of the three measures (i.e., precision, recall, and F-score), when evaluating the traceability links between the source and target artifacts. The second criterion is the comparison of the proposed trace retrieval approach with two experiments (IR and TraceLab) as shown in Tables IV (A) to (C) and two studies ([19]and [24]) as shown in Tables V (A) to (C). The proposed trace retrieval approach achieved high results compared to the two experiments and the two studies.

VI. CONCLUSION AND FUTURE WORK

This study presented new clustering-based approach that addresses the term mismatch problem to obtain the greatest improvements in precision. The study followed the proposed research direction toward realizing automated trace retrieval by developing intelligent tracing solutions. Then the authors applied the intelligent solution that is based on unsupervised learning using clustering. After that, evaluate the proposed approach results with respect to two criteria: performance of the confusion matrix (i.e., precision, recall, F-score) and comparison of the proposed trace retrieval approach with two experiments and two studies. Clustering yields high results in precision with other measures (i.e., recall, F-score, and accuracy), which is a big concern associated with trace retrieval precision.

In the future work, the authors will look for another intelligent solution that can be applied to the same datasets (i.e., eTour, SMOS, and eANCI) for improving trace retrieval precision.

REFERENCES

[1] A. Kannenberg, & H. Saiedian, why software requirements traceability remains a challenge. CrossTalk The Journal of Defense Software Engineering, (2009). 22(5), 14-19).

[2] N.Al-walidi, A. Khamis, & N. Ramadan, "Systematic Literature Review of Recommender Systems for Requirements Engineering", The International Journal of Computer Applications (IJCA) (2020).

[3] J. Cleland-Huang, O.Gotel, C. Huffman Hayes, J. P Mäder,& A. Zisman, Software traceability: trends and future directions. In Future of software engineering proceedings (2014). (pp. 55-69).

[4] D.Hanspeter, A.Janes, A.Sillitti, & G. Succi, Improving the identification of traceability links between source code and requirements. In DMS (2012). (pp. 95-100).

[5] X.Zou, R. Settimi, & J. Cleland-Huang, improving automated requirements trace retrieval: a study of term-based enhancement methods. Empirical Software Engineering, (2010) 15(2), 119-146.

[6] H. Kim.Unsupervised learning. In Artificial Intelligence for 6G (2022). (pp. 35-86). Springer, Cham.

[7] A. Ayed, B. M. B.Halima, & AM.limi, Survey on clustering methods: Towards fuzzy clustering for big data. In 2014 6th International conference of soft computing and pattern recognition (SoCPaR) (2014, August) (pp. 331-336). IEEE.

[8] R. Petegrosso, Z. Li, & R. Kuang, Machine learning and statistical methods for clustering single-cell RNA-sequencing data. Briefings in bioinformatics, (2020)21(4), 1209-1223.

[9] A. Rajabi, M. Eskandari, M. J. Ghadi, L. Li, J. Zhang, & P. A Siano, comparative study of clustering techniques for electrical load pattern segmentation. Renewable and Sustainable Energy Reviews, (2020). 120, 109628.

[10] S.Babichev, V.Lytvynenko, &V. Osypenko, Implementation of the objective clustering inductive technology based on DBSCAN clustering algorithm. In 2017 12th international scientific and technical conference on computer sciences and information technologies (2017, September). (csit) (Vol. 1, pp. 479-484). IEEE.

[11] X. Zou, R.Settimi, &J. Cleland-Huang, Improving automated requirements trace retrieval: a study of term-based enhancement methods. Empirical Software Engineering, (2010). 15(2), 119-146.

[12] N. Ali, H. Cai, H., Hamou-Lhadj, A. J, & Hassine,. Exploiting Parts-of-Speech for effective automated requirements traceability. Information and Software Technology, (2019). 106, 126-141.

[13] D. Cuddeback, A.Dekhtyar, & J.Hayes, Automated requirements traceability: The study of human analysts. In 2010 18th IEEE International Requirements Engineering Conference (2010, October). (pp. 231-240). IEEE.

[14] C. Mills, J. Escobar-Avila, & Haiduc, S. Automatic traceability maintenance via machine learning classification. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME) (2018, September). (pp. 369-380). IEEE.

[15] R. Lapeña,J . Font,C. Cetina, &O. Pastor, (2018, June). Exploring new directions in traceability link recovery in models: The process models' case. In International conference on advanced information systems engineering (pp. 359-373). Springer, Cham.

[16] W.A. Zogaan, (2019). Towards an Intelligent System for Software Traceability Datasets Generation. Rochester Institute of Technology.

[17] A.I. Montoya-Munoz, &O. M. C. Rendon, (2020). An approach based on fog computing for providing reliability in iot data collection: A case study in a colombian coffee smart farm. Applied Sciences, 10(24), 8904.

[18] J. O. Palacio-Niño, & F.Berzal, (2019). Evaluation metrics for unsupervised learning algorithms. arXiv preprint arXiv:1905.05667.

[19] T.Du, B. Shen, G. H. Huang, Z. Q. Y. S. Yu, , & Wu, D. X. Automatic traceability link recovery via active learning. Frontiers of Information Technology & Electronic Engineering, (2020) 21(8), 1217-1225.

[20] J.Lin, Y. Liu, , Q. Zeng,M. Jiang, &J. Cleland-Huang, Traceability transformed: Generating more accurate links with pre-trained bert models. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (2021, May). (pp. 324-335). IEEE.

[21] H. Wang, Shen,Z . Huang,Y.Yu, &K. Chen, Analyzing close relations between target artifacts for improving IR-based requirement traceability recovery. Frontiers of Information Technology & Electronic Engineering, (2021). 22(7), 957-968.

[22] Y.Li, &J. Cleland-Huang, Ontology-based trace retrieval. In 2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE) (2013, May). (pp. 30-36). IEEE.

[23] J. Guo,N . Monaikul, C.Plepel, &J. Cleland-Huang, Towards an intelligent domain-specific traceability solution. In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (2014, September). (pp. 755-766).

[24] T.Li, S. Wang, D. Lillis, &Z. Yang, combining machine learning and logical reasoning to improve requirements traceability recovery. Applied Sciences, (2020). 10(20), 7253.

[25] P. Sharma, (2017). Datasets Used in Fifteen Years of Automated Requirements Traceability Research. Rochester Institute of Technology.

[26] F. Faiz, R.Easmin, &A.U.Gias, (2016). Achieving Better Requirements to Code Traceability: Which Refactoring Should Be Done First? 2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC). doi:10.1109/quatic.2016.012

[27] CoEST: Center of excellence for software traceability, http://www.CoEST.org.

[28] C. Mills,J.Escobar-Avila, Bhattacharya, A.,Kondyukov, G.S. Chakraborty, &S. Haiduc, (2019, September). Tracing with less data: active learning for classification-based traceability link recovery. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 103-113). IEEE.

[29] R. Oliveto, M.Gethers,D. Poshyvanyk, &A. De Lucia, On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery. 2010 IEEE 18th International Conference on Program Comprehension. (2010). doi:10.1109/icpc.2010.20.

[30] D. Falessi,M . Di Penta,G. Canfora,&G. Cantone, (2017). Estimating the number of remaining links in traceability recovery. Empirical Software Engineering, 22(3), 996-1027.

[31] A. E.Ezugwu,A.M. Ikotun,O.O. Oyelade,L. Abualigah,L.O. Agushaka, , C.I.Eke, &A.A. Akinyelu, (2022). A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Engineering Applications of Artificial Intelligence, 110, 104743.

[32] H. Bahri, & T. S. T. Mahadi, (2016). Google translate as a supplementary tool for learning Malay: A case study at Universiti Sains Malaysia. Advances in Language and Literary Studies, 7(3), 161-167.

[33] D. Khyani, B. S.Siddhartha, N. M. Niveditha, , & B. M. Divya, (2021). An Interpretation of Lemmatization and Stemming in Natural Language Processing. Journal of University of Shanghai for Science and Technolog

[34] A. Abubakar, I.A. Roko, A.B. Muhammad, &I. Saidu, Hausa WordNet: An Electronic Lexical Resource. Saudi Journal of Engineering and Technology, (2019). 4(8), 279-285.

[35] M. Khatun, Evaluating Word Embedding Models for Traceability (Doctoral dissertation, Louisiana State University and Agricultural & Mechanical College) (2021).

[36] N. Alsuraiby, &S. Albarrak, Cosine similarity-based algorithm for social networking recommendation. International Journal of Electrical & Computer Engineering (2022). (2088-8708), 12(2).

[37] B. Dit,E . Moritz, M. Linares-Vásquez,D. Poshyvanyk,&J. Cleland-Huang, Supporting and accelerating reproducible empirical research in software evolution and maintenance using tracelab component library. Empirical Software Engineering, (2015). 20(5), 1198-1236.

[38] D.Farrar, & J. H. Hayes, A comparison of stemming techniques in tracing. In 2019 IEEE/ACM 10th International Symposium on Software and Systems Traceability (SST) (2019, May). (pp. 37-44). IEEE.

[39] J.Payne, & J. H.Hayes, (2019, May). University of kentucky tracelab component similarity matrix voting merge. In 2019 IEEE/ACM 10th International Symposium on Software and Systems Traceability (SST) (pp. 17-20). IEEE.

[40] http://selab.netlab.uky.edu/homepage/pages/TraceLabCatalogue/TracelabCatalogue/componentspage.html(2018).