Received: July 26, 2023. Revised: September 3, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Enhancing Recovery Links between Two Artifacts Using Variational Autoencoder

Nejood Hashim Al-walidi[1]* **Nagy Ramadan Darwish[1]**

[1]*Department of Information Systems and Technology,*
Faculty of Graduate Studies for Statistical Research, Cairo University, Cairo, Egypt
* Corresponding author's Email: nejood96@yahoo.com

Abstract: Traceability link recovery is a crucial task in software engineering that ensures the development of dependable and credible software systems. Traceability links between requirements and source code support various activities in the software development process, including change management and software maintenance. These links can be established manually or automatically. Manual trace retrieval is a time-consuming task. Automatic trace retrieval can be performed via various tools such as information retrieval or machine learning methods. Some automatic tools couldn't retrieve the links between requirements and source code. 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 low precision problem caused by the term mismatch problem between requirements and source code to obtain the greatest improvements in trace retrieval accuracy. The proposed approach utilizes a variational autoencoder (VAE), an unsupervised deep-learning model in the automated trace retrieval process. We have conducted a series of experiments on three datasets: eTour, SMOS, and eANCI to evaluate our approach against existing approaches. In order to validate the effectiveness of our proposed approach, we compared it to three previous studies that addressed the same problem and utilized the same datasets: the first study used unsupervised machine learning based on clustering, the second study used active learning, and the third study used a classification machine learning. The results show that our proposed approach improves the trace retrieval precision in the automated trace retrieval process.

Keywords: Requirements traceability, Term mismatch, Trace retrieval, Deep learning, Variational autoencoder.

1. Introduction

Traceability link recovery (TLR) is a crucial task in software engineering that involves restoring links between source artifacts (such as requirement documents) and target artifacts (such as source code) within the same project [1]. Traceability is a critical aspect of software development [2] and maintenance, as it supports various activities, including program comprehension, compliance verification, change impact analysis, and regression analysis of test cases [3].

Traceability links can be established manually or automatically. However, manual retrieval of traceability links can be error-prone [4] and time-consuming. Therefore, automatic retrieval techniques that utilize tools such as information retrieval [5], ontology, machine learning [6], and deep learning [7] are often employed. Deep learning approaches can be classified into two main categories: supervised learning and unsupervised learning [8]. Deep learning has exhibited impressive performance across multiple domains, with particular success in tasks related to natural language processing (NLP) [9].

One major issue in trace retrieval research is the problem of low precision. Precision refers to the proportion of accurate traces among all the retrieved traces. When precision is low, it means that numerous false traces are being incorrectly retrieved, requiring users to manually evaluate the retrieved links to identify the correct traces. This highlights the need to focus on improving precision. Consequently, several researchers have explored various approaches to

enhance precision [6, 10]. However, these attempts have only yielded marginal improvements. The primary cause of this problem lies in term mismatches across the documents to be traced.

A term mismatch problem can arise between source and target artifacts when the language used in the target document neither matches the language of the source document nor matches project-level synonyms defined in a project glossary [11]. The problem is that datasets have a limited number of labels. These labels are provided for testing only. We formulated the problem by choosing an intelligent solution based on unsupervised deep learning using a variational autoencoder. The suggestion for addressing this problem is to choose an intelligent solution based on unsupervised deep learning to find probabilities among data points and effectively group those that are similar, ultimately enhancing trace retrieval precision. Therefore, the reason for choosing the proposed approach is to enhance trace retrieval precision, and this solution is suitable for data that has few labels or unlabelled data.

In this study, to address the term mismatch problem in automated trace retrieval, we follow the proposed research direction toward achieving automated trace retrieval by developing an intelligent tracing solution namely unsupervised deep learning based on VAE.

The variational autoencoder (VAE) is an unsupervised deep learning model designed to handle unlabelled datasets or a few labelled datasets, meaning datasets that do not have direct class labels associated with the data instances [12]. The VAE is a popular and effective model applied to text modelling for generating various sentences [13] and learning representations of high-dimensional data [14]. The primary advantage of the VAE is for learning smooth potential state representations of input data. VAE, known as variational autoencoder, is a latent variable model based on probability. The observed vector $\boldsymbol{x}$ exhibits a correlation with the low-dimensional latent variable $\boldsymbol{z}$ through a conditional distribution [15]. VAE is widely used in many natural language processing (NLP) tasks, such as text modelling [16]. VAE has two components encoder and decoder. VAE simulates the probability of x as shown in Eq. (1).

$$P_\theta(x) = \int P_\theta(x|z)\, P_\lambda(z)\, dz| \quad (1)$$

In this context, the posterior probability of x given **z**, denoted as $P_\theta(x|z)$, is represented by a neural network with parameters θ. Similarly, the prior probability of the latent variables, denoted as $p\lambda(z)$, is modelled by neural networks with parameters λ. These neural networks are referred to as decoders.

As proposed, the encoder and decoder are essential components of the VAE and operate collaboratively to learn a compressed representation (i.e., a latent space) of the input (service descriptions). To evaluate the reconstruction performance and ensure that the latent space captures meaningful features of inputs. The third essential part of the VAE is the cost function, which is employed [17]. Deep recurrent networks such as ANN and LSTM are often utilised to implement the encoder and decoder in text modelling [18].

This paper utilizes four commonly utilized metrics: precision, recall, F-score, and accuracy, to assess the efficacy of the newly suggested method. The evaluation of requirement tracing tools' effectiveness is based on precision and recall [19].

Precision refers to the proportion of accurately retrieved candidate links expressed as a percentage as shown in Eq. (2), It measures the relevance of retrieved documents [20]. Recall refers to the correctly identified links [21] percentage as shown in Eq. (3). The F-score utilizes to provide a balance between precision and recall. It is the harmonic mean of precision and recall [22] percentage as shown in Eq. (4). Lastly, accuracy measures the percentage of correctly classified normal and the outlier values among the total numbers of the classifications percentage as shown in Eq. (5). Both precision and recall are essential. The F-Score is usually preferred for evaluating trace results where recall is considered more significant than precision [23].

$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN) \quad (2)$$

$$\text{Precision} = TP/ (TP + FP) \quad (3)$$

$$\text{Recall} = TP/ (TP + FN) \quad (4)$$

$$\text{F-Score} = 2*(\text{precision}*\text{recall})/(\text{precision} + \text{recall}) \quad (5)$$

The study is structured as follows: Section 2 presents related work in trace retrieval; Section 3 outlines the proposed approach; Section 4 discusses the obtained results; Section 5 evaluates the results; and section 6 concludes with a discussion of future work.

2. Related studies

This section reviews some methods used to solve the term mismatch problem in automated trace retrieval between requirements and source code.

In [6], the authors proposed an approach that addresses the term mismatch problem between requirements and source code to obtain the most significant improvements in trace retrieval accuracy. The proposed approach used unsupervised machine learning based on the clustering in the automated trace retrieval process and performed an experimental evaluation against previous benchmarks. Unfortunately, this proposed approach has trouble identifying some links between requirements and source code that are very similar to those found in standard groups, which might lead the items to the wrong cluster.

The authors suggested a framework: BERT (T-BERT) for establishing trace links between requirements and source code. They employed this framework to restore the link between issues and commits in open-source projects. The evaluation of three BERT architectures revealed that the single-BERT architecture produced the most precise links, while the Siamese-BERT architecture delivered similar outcomes with notably lesser execution time. By gaining knowledge and transferring it, all three models in the framework surpassed classical IR trace models in terms of accuracy and efficiency. The results show that the single-BERT architecture generated the most accurate links, while the Siamese-BERT architecture produced comparable results with significantly less execution time. Unfortunately, their method is a conventional way of leveraging OSS projects for traceability which may affect the true links may be missed. Additionally, the authors didn't use precision, which is considered an essential measure for recovery links between requirements and source code [24].

To overcome the limitation of IR/ML techniques and adopt a probabilistic perspective towards the traceability problem, the authors in [25] designed and implemented a HierarchiCal PrObabilistic model for software traceability (COMET) that can infer candidate trace links. This model used a hierarchical Bayesian network to model the presence of traceability links. The authors demonstrated that COMET outperforms IR/ML techniques and has considerable potential for industrial applications. They intend to extend the application of COMET to new information sources, customize its analysis to deduce security-related links, and deploy the COMET plugin in collaboration with industry partners to gather feedback. The drawback of their model is that suitable for small systems, so it's difficult to generalize it to other systems to infer a set of candidate trace links [25].

Guo and his co-authors [11] examined and compared three methods for enhancing queries to address the term mismatch problem and improve the quality of trace links between regulatory codes and requirements. The first technique involves training a classifier to replace the original query with terms learned from a training set of trace links between regulations and requirements. The second method replaced the original query with terms obtained through web mining, while the third uses a domain ontology to expand query terms. The ontology created manually, using an approach that leverages existing traceability knowledge. To evaluate the effectiveness of these techniques, they applied them to trace security regulations from the USA government's health insurance privacy and portability act (HIPAA) against ten healthcare-related requirements specifications. Their results show that the classification-based approach yielded the best results, but improvements were also observed with both the classification and ontology-based solutions. On the negative side, the web-mining technique showed improvements only in some queries. Web mining and ontology techniques achieved lower results which means those techniques didn't have the ability to retrieve all the correct links between requirements and source code.

3. The proposed approach

In this section, the authors divided the task of the proposed approach using the VAE model to improve the recovery of links between requirements and source code into four main phases. Fig. 1 illustrates the four phases of the proposed trace retrieval approach based on VAE. The four phases are as follows:

Phase one is choosing suitable. Phase two includes seven steps (feature extraction, translation, preprocessing, lemmatization, query expansion, TFIDF, and SMOTE). Phase three contains four steps (encoder, decoder, train VAE, and dense NN layer). Phase four is the evaluation method of the retrieved links between requirements and source code as shown in Fig. 1.

Phase 1: Choosing suitable datasets

Research in the area of automated requirements traceability relies on the availability of different types of datasets. Obtaining such datasets has been one of the reported barriers by researchers in the software engineering domain [26]. This phase introduces the three datasets that are used in this study to evaluate the automated trace retrieval between source artifact (requirements) and target artifact (source code).

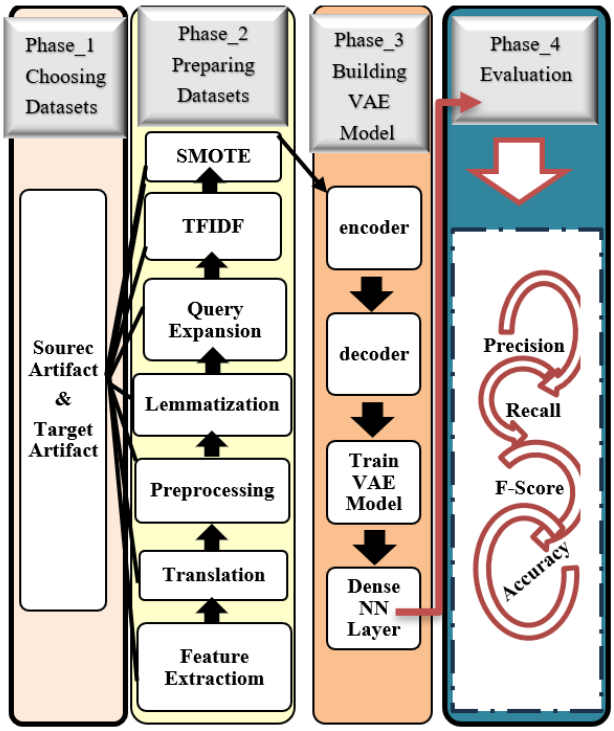


Figure. 1 Proposed trace retrieval approach using VAE

Table 1. Characteristics of the Three Datasets [26, 27, 28, 29, 30]

Name	Description	Source Artifact	Target Artifact	Correct Links
eTour	"Is an electronic tourist guide developed by students"	Use Cases (58)	Source Code Classes (116)	336
SMOS	"Is an application that is used to monitor high school students (e.g., absence, grades)."	Use Cases (67)	Source Code Classes (100)	1045
eANCI	"A system providing support to manage Italian municipalities"	Use Cases (140)	Source Code Classes (55)	567

Table 1 briefly defines the characteristics of the datasets used in the study approach. These datasets are available through http://www.CoEST.org.

The reason for choosing these datasets is because they are publicly available, meaning that anyone can access them and use them for research or analysis this is important because it allows for greater transparency. Also, these datasets are used to evaluate the automated trace retrieval approach between requirements and source code. The three datasets are compatible with different tools such as Python.

Phase 2: Preparing the datasets.

The authors divided this phase into seven steps as follows:

Step 1, Feature extraction. In order to establish traceability links between the source artifact (written in natural language) and the target artifact (written in the programming language Java) was necessary to extract key features from each. The source artifact contained various types of information, but not all of it was relevant to the study's goal. Thus, the authors focused on extracting the most significant features from the source code, which included the *class name, class attributes, class comments, method comments, method name, method parameters, and method return*. They also extracted two primary features from the requirements: title and description.

Step 2, Translation. Requirements and source code artifact contained some Italian text, including words and sentences within certain lines. As a result, the authors needed to translate the content of both artifacts into English. To accomplish this, authors utilized a sophisticated translation engine to translate the documents into English, especially if they were initially not written in English. Specifically, they employed ChatGPT. "ChatGPT is a public tool developed by OpenAI that is based on the GPT language model technology"[31]. "ChatGPT is a single model handling various NLP tasks and covering different languages, which can be considered a unified multilingual machine translation model." [32].

Step 3, Applying other pre-processing steps, such as camel_case_split, under_score_split, converting from upper case to lower case, removing punctuations, removing stop words, removing numbers, and removing single characters.

Step 4, Lemmatization. Lemmatization is a fundamental NLP task that includes transforming a word in its inflected form to its base or lemma form. It is a technique employed in a variety of NLP tasks. The primary objective of lemmatization is to standardize words to their canonical forms, which can enhance the accuracy and effectiveness of subsequent NLP tasks [33].

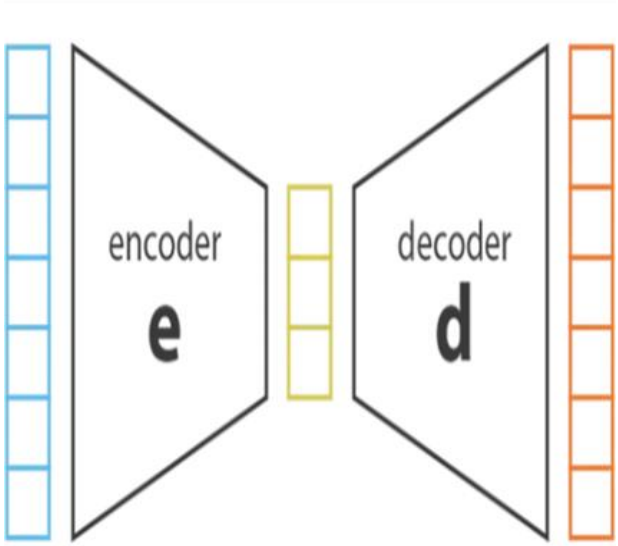


Figure. 2 VAE Architecture [41]

Step 5, Query expansion (QE). Query expansion is a method of finding suitable terms for the reformulation of queries to resolve the short query and word mismatch problem and enhance the performance of the retrieving the information [34]. WordNet is a lexical database for the English language that is commonly used in natural language processing and computational linguistics. It is a large-scale, electronic database that contains information about words and their semantic relationships, such as synonyms, antonyms, hypernyms (words that are more general), hyponyms (words that are more specific), and meronyms (words that are part of the whole) [35]. In this step, the authors employed QE using WordNet to augment queries with related words, with the goal of increasing the number of retrieved documents and enhancing recall performance. To achieve this, the researchers extracted all words within each query and automatically selected synonyms for each individual word.

Step 6, Encoding (TFIDF). There are several techniques available to transform the text into numerical vectors, such as TFIDF encoding, Dec2Vec, Word2Vec, and bag of words (BOW) [36]. In this study, authors opted to use TFIDF encoding as it was found to produce better results compared to other methods that they experimented with in their previous study.

Step 7, SMOTE method. The authors previously acknowledged certain problems associated with three datasets (eTour, SMOS, and eANCI) in their research (See ref. [6]). Specifically, all three datasets are imbalanced, and to address this issue, the authors proposed oversampling algorithm for deep learning models called SMOTE, which based on the widely used SMOTE method. The authors applied this method before training the VAE model.

Phase 3: Build the variational autoencoder (VAE) model

Variational autoencoders (VAEs) are generative models that learn to simulate the latent representation of data [37]. It is used to improve the quality of the generated outputs [38]. Variational autoencoders (VAEs) [39] consist of two deep neural networks: an encoder network and a decoder network [40]. Fig. 2 illustrates the architecture of the VAE mode.

This phase consists of four steps as follows:

Step 1. Creating the encoder model. Given an input data point **x**, the VAE aims to encode it into a low-dimensional latent space representation z. These are achieved through an encoder network that parameterizes a distribution **qφ(z|x),** where **φ** represents the encoder network's parameters [40].

Step 2. Creating the decoder model. The latent variable **z** is then passed through a decoder network, parameterized by **θ,** which aims to reconstruct the original input **x**. The decoder outputs the parameters of the conditional distribution P_{θ} **(x|z),** which represents the reconstructed data distribution [42].

The process of encoding involves the acquisition of latent variables from the input, and the decoding process generates output by utilizing samples of these latent variables with an adequate amount of training data both the encoder and the decoder can be trained simultaneously by minimizing the reconstruction loss and the Kullback-Leibler (KL) divergence between the distributions of the latent variables and independent normal distributions [40].

Generally, the encoder network turns the input samples (**x**) into two parameters in a latent space: (**z_mean and z_log_sigma**). Then, it randomly samples similar points **(z)** from the latent normal distribution that is assumed to generate data via **z = z_mean + exp(z_log_sigma) * epsilon**, where epsilon is a random normal tensor. Finally, the decoder network maps these latent space points back to the original input data.

Custom loss function: The loss function consists of two components. The first component is the reconstruction error **Eqφ(z|x) [log P θ (x|z)].** The second component is the relative entropy, which facilitates learning of the model distribution **qφ(z|x)** to approximate the actual prior probability **pλ(z),** specifically a normal distribution. Before training the variational autoencoder model, the final step is to create a custom loss function and compile the model. Then, the loss function will compare inputs and outputs and try to minimize the difference between them [43].

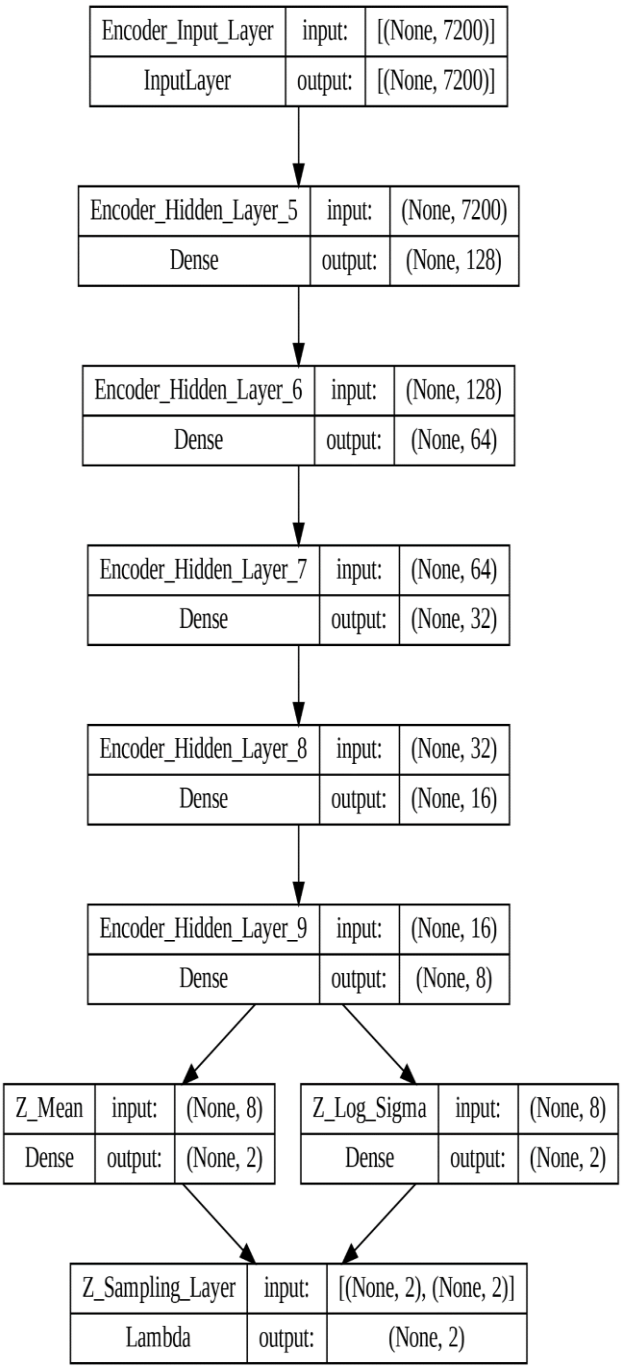


Figure. 3 Creating the encoder model (structural diagram)

Step 3: Training the VAE model. The overall objective is to minimize the sum of the reconstruction loss and the regularization loss. These are done by optimizing the parameters of both the encoder **(φ)** and the decoder **(θ)** networks using techniques such as stochastic gradient descent (SGD) or its variants [44].

The authors specified dimensions for input/output and latent space layers. They determined the original_dim is equal to 7200, and the latent space dimension is equal to 2, as shown in Fig. 3. The authors trained the VAE model for 150 epochs with a batch size of 16 in the experiments as follow:

```
history = vae.fit(x_train, x_train, epochs=150, batch_size=16, validation_data=(x_val, x_val))
```

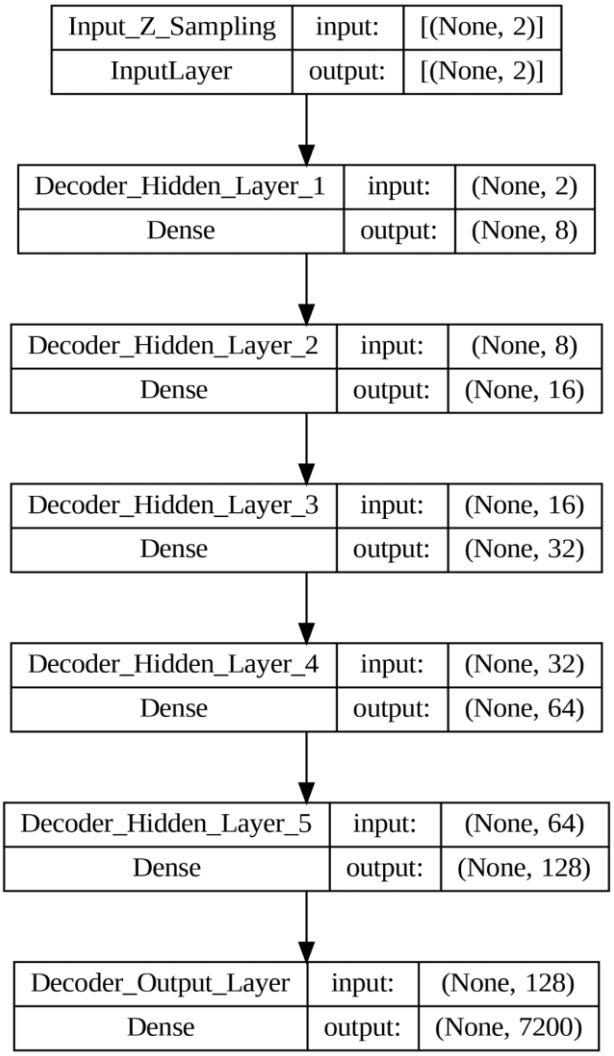


Figure. 4 Creating the decoder model (structural diagram)

Fig. 4 shows the processes of creating a decoder model. It is an essential component of VAE. It's responsible for mapping an encoded latent space representation of the input data back to its original form. The figure illustrates the steps involved in constructing the decoder model architecture.

Step 4. Dense NN layer

In this step, a dense layer in a neural network is characterized by its fully interconnected nature with the preceding layer, where each neuron in the layer is connected to every neuron of its preceding layer. This layer is widely employed in artificial neural networks and is considered as one of the most commonly used layers [45]. To determine the most likely answer to a classification problem, the Sigmoid activation

Table 2. Results of the proposed trace retrieval approach using VAE

Datasets	Precision	Recall	F-Score	Accuracy
eTour	0.93	**0.99**	**0.96**	**0.99**
SMOS	0.91	**0.99**	0.95	0.98
eANCI	**0.95**	0.91	0.93	0.97

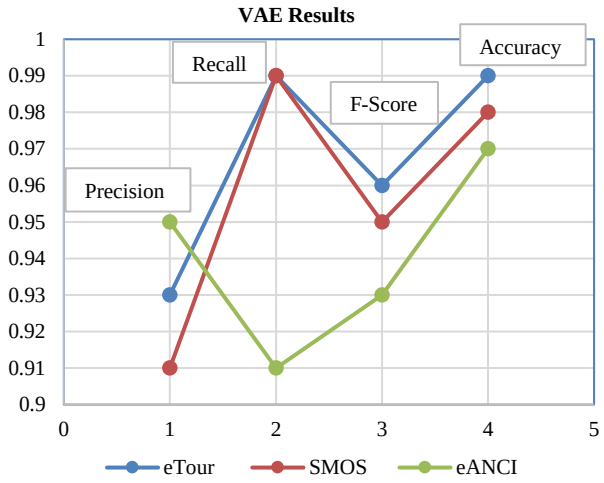


Figure. 5 The Results of the proposed trace retrieval approach using VAE

function is employed at the output layer [46] and maps the input to the range between (0 and 1).

clf = Dense (1, activation='sigmoid') (vae. Layers [1], output [2])

Phase 4: Evaluation

In this phase, the evaluation method of the retrieved links between requirements and source code using VAE is the performance of the four measures (i.e., precision, recall, F-score, and accuracy).

4. Results

Before presenting the results, we will evaluate the success of the strategy provided in this study. We have used Python Google colab as a cloud-based platform that allows users to write and execute Python code in a web browser without requiring any setup. Various types of Python packages are used, such as the scikit-learn package, TensorFlow, and Keras platform. We set the hyperparameters to train the VAE model. First, we specified the original_dim = 7200 and the latent space dimension= 2, the epochs= 150, and batch_size=16. The results of those hyperparameters are as follows: *the total params: 932,767, trainable params: 932,767, and non-trainable params: 0*. Also, we placed the epochs=20 and batch_size=10 to fit the VAE model to get optimal parameters for the encoder and decoder networks. The encoder encodes the input data into a probabilistic distribution in a lower-dimensional latent space. The mean (μ) and log-variance (log 2) parameters define this distribution. Latent variables are sampled through reparameterization, and the original data is then decoded using a generator network. The regularisation term (KL divergence) in the VAE's loss function maintains the latent space's resemblance to a typical Gaussian distribution. These elements are optimised during training, allowing the VAE to produce unique data by sampling from the latent space. Second, we compute the confusion matrix: precision, recall, F-score and accuracy using " *sklearn.metrics*" to get the final results.

The results of this study are divided into two parts as follows:

First: Results of applying the proposed trace Retrieval Approach based on VAE described in Section 3, which consists of four phases, as shown in Fig. 1. Second: comparing the proposed trace retrieval approach results with three studies (i.e., studies [6] and [47], and [47]). Study ([6]) utilized unsupervised machine learning-based clustering, Study ([47]) utilized active learning, and study ([48]) utilized supervised machine learning.

First: Results of the proposed trace retrieval approach based VAE.

As mentioned in section 3, the authors applied the Variational autoencoder to the three datasets (i.e. eTour, SMOS, and eANCI) described in Phase 1 of the proposed trace retrieval approach. The results of the proposed trace retrieval approach based on VAE are presented here, which include four measures: Precision, recall, F-Score, and accuracy, as shown in (Table 2) and Fig. 5. The proposed approach achieved the highest results using the four measures of the confusion matrix with the three datasets, as shown in (Table 2) and Fig. 5.

Table 2 and Fig. 5 indicate that the VAE achieves the highest results across three datasets, with some variation between the four measures. For instance, the highest precision is achieved with the eANCI dataset, the highest recall across eTour and SMOS datasets, and the highest F-score and accuracy with the eTour dataset.

Based on Table 2, these findings suggest that the VAE-based approach is effective across different datasets and has the ability to perform well in multiple evaluation measures. However, the variation between the four measures indicates that the approach may excel in certain areas over others depending on the dataset used. This highlights the importance of considering the unique characteristics of each dataset when evaluating the performance of the approach. Overall, the results demonstrate the potential of the VAE for improving precision, recall, F-score, and accuracy.

Table 3. Results comparison: VAE and Clustering: (a) eTour, (b) SMOS, and (c) eANCI

(a)

Measure	VAE	Clustering
Precision	**0.93**	**0.93**
Recall	**0.99**	0.97
F-Score	**0.96**	0.94
Accuracy	**0.99**	0.91

(b)

Measure	VAE	Clustering
Precision	**0.91**	0.73
Recall	**0.99**	0.76
F-Score	**0.95**	0.74
Accuracy	**0.98**	0.66

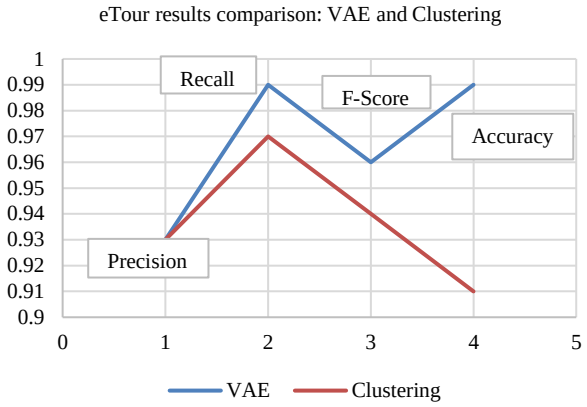
(c)

Measure	VAE	Clustering
Precision	**0.95**	0.64
Recall	**0.91**	0.77
F-Score	**0.93**	0.70
Accuracy	**0.97**	0.60

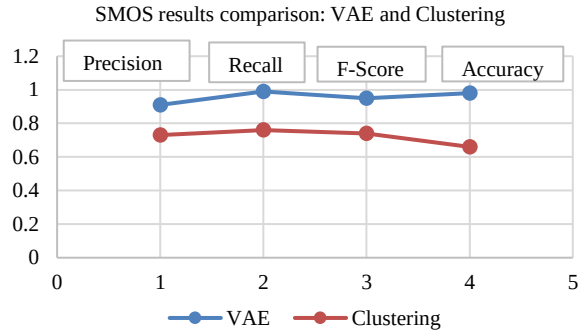
Second: We compared the proposed trace retrieval approach results with three previous studies (i.e., studies [6] and [47], and [48]) and using three Parameters of comparison (i.e., precision, recall, and F-score) and these parameters are used with our proposed approach also were have used in three previous studies. Our proposed approach and three previous studies used the same datasets (i.e., eTour, SOMS, and eANCI).

Study ([6]) utilized unsupervised machine learning based on clustering to address the same problem and used the same datasets for solving low precision between requirements and source code. The authors applied four clustering algorithms (K-means++, GMM, Hierarchical, and DBSCAN) to three datasets (eTour, SMOS, and eANCI). We have chosen the highest results from this study to compare with our proposed approach. Table 3 (a) to (c) presents a comparison between the proposed trace retrieval approach based on VAE and unsupervised machine learning based on clustering using the K-means++ algorithm. Four metrics, namely precision, recall, F-score, and accuracy, are used to compare the performance of these approaches.

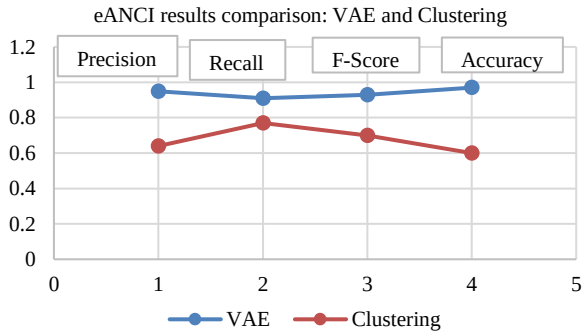
Tables 3 (a) to 3(c) and Fig. 6 (a) to (c) depict that the proposed trace retrieval-based approach, based on VAE, achieved the same precision results as the study ([6]) in the eTour dataset. In contrast, VAE achieved higher results in recall and F-score in the eTour dataset compared to the study ([6]). In the SMOS and eANCI datasets, the proposed trace retrieval approach based on VAE achieves the highest results in precision, recall, F-score, and accuracy compared to the study ([6]).



(a)



(b)



(c)

Figure. 6 Result comparison: VAE and Clustering: (a) eTour, (b) SMOS, and (c) eANCI

Table 4 (a to c) presents a comparison between the two studies (Study [47] and Study [48]) and the proposed approach based on VAE.

In (Study [47]), the authors proposed an approach for trace link recovery (TLR) based on active learning (AL), referred to as the AL-based approach to retrieve links between requirements and source code. We have chosen the highest results from this study to compare with our proposed approach. In (Study [48]), the authors introduced an approach for traceability link recovery, which focuses on measuring the similarity between requirements and source code by exploring their respective features. To achieve this,

Table 4. Results comparison: VAE and two Studies: (a) eTour, (b) SMOS, and (c) eANCI

(a)

Measure	VAE	Study [47]	Study [48]
Precision	**0.93**	0.68	0.66
Recall	**0.99**	0.34	0.59
F-Score	**0.96**	0.46	0.61

(b)

Measure	VAE	Study [47]	Study [48]
Precision	**0.91**	0.57	0.75
Recall	**0.99**	0.29	0.33
F-Score	**0.95**	0.39	0.51

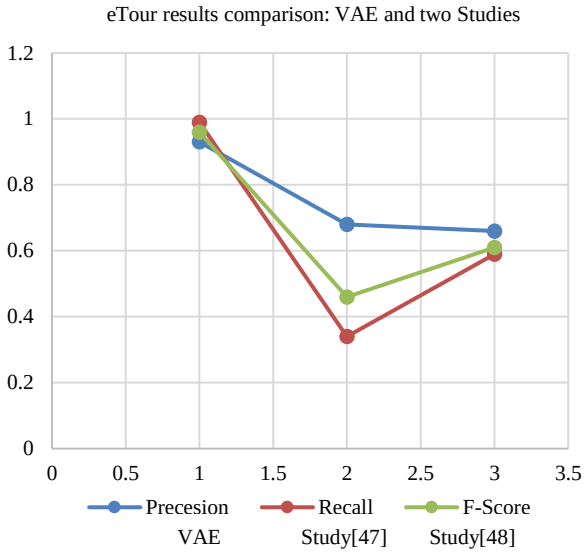
(c)

Measure	VAE	Study [47]	Study [48]
Precision	**0.95**	0.73	0.62
Recall	**0.91**	0.44	0.54
F-Score	**0.93**	0.55	0.58

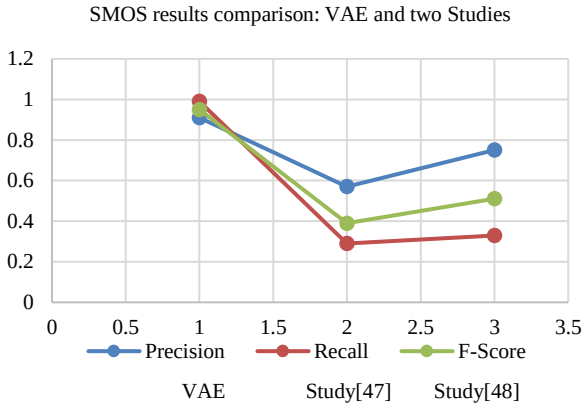
they combined machine learning and logical reasoning models. We have chosen the highest results from this study to compare with our proposed approach.

The following Tables 4 (a) to 4(c) present the results of the comparison between our proposed approach using VAE and (Study [47], and Study [48]) using three parameters (i.e., precision, recall, and F-score).

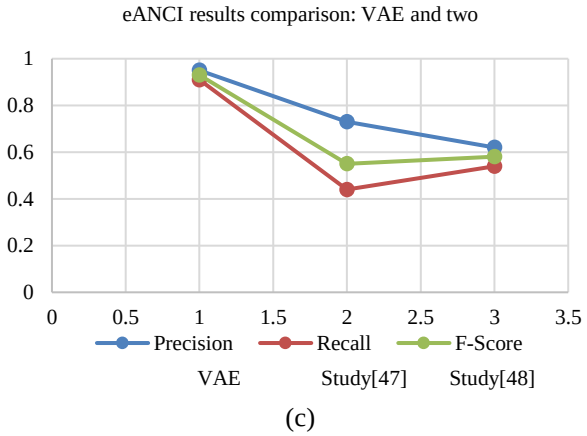
Tables 4 (a) to (c) and Fig. 7 (a) to (c) present the results of the evaluation conducted on the proposed trace retrieval approach, which is based on the variational autoencoder (VAE). The evaluation was performed using three datasets, and the primary focus of this study was the precision measure. However, the proposed approach also demonstrated high performance in terms of recall and F-score. Comparing the results with two other studies referenced (Study [47] utilizing active learning techniques) and (Study [48]) combining machine learning and logical reasoning models, it is evident that the proposed trace retrieval approach achieved superior results across all three evaluation measures: precision, recall, and F-score. The studies [47] and [48], despite employing the same evaluation measures, obtained lower results when compared to the VAE-based approach. This indicates that the VAE-based approach outperformed these alternative methods in terms of precision, recall, and F-Score. The superiority of the VAE-based approach can be attributed to its effectiveness in retrieving relevant traces while maintaining a balance between precision (the proportion of retrieved traces that are relevant) and recall (the proportion of relevant traces that are retrieved). The F-score, which combines precision and recall into a single metric, further reinforces the strong performance of the VAE-based approach.



(a)



(b)



(c)

Figure. 7 The comparison between VAE, Study ([47]), and Study ([48]): (a) eTour Results, (b) SMOS Results, and (c) eANCI Results

Overall, the findings presented in Tables 4 (a) to 4(c) provide robust evidence supporting the effectiveness of the proposed trace retrieval approach based on VAE. When compared to the alternative approaches explored in studies ([47] and [48]), the VAE-based approach demonstrates significant improvements

across all evaluation measures, highlighting its potential for accurate and reliable trace retrieval.

According to the above results, the suitable intelligent solution for addressing the low precision caused by term mismatch between source artifacts (such as requirement documents) and target artifacts (such as source code) for a few labels or unlabelled data is unsupervised deep learning using VAE.

5. Evaluation

In this section, the authors describe the evaluation process of the retrieved links based on two criteria. The initial criterion assesses the efficacy of the four measures (i.e., precision, recall, F-score, and accuracy) in evaluating the traceability links between the source and target artifacts. The second criterion involves comparing the proposed trace retrieval approach based on unsupervised deep learning using VAE with three previous studies (study [6], study [47], and study [48]) as shown in Tables 3 (a) to (c) and Table 4 (a) to (c).

6. Conclusion and future works

This paper proposed an intelligent solution based on unsupervised deep learning using VAE that addresses the term mismatch problem between requirements and source code to obtain the greatest improvements in precision and other measures. The proposed approach includes four phases described in section 3. Each phase has its own steps, as shown in Fig. 1. The proposed approach was evaluated using two criteria: the performance of the confusion matrix, which includes precision, recall, F-score, and accuracy, as well as a comparison of results of our proposed approach against three previous studies [(study [6], study [47], and study [48])]. In the first study [(6)], we utilized four metrics (precision, recall, F-score, and accuracy) for comparison with our proposed approach. In the second study [(47)] and the third study [(48)], we employed three metrics (precision, recall, and F-score) for comparison with our proposed approach. The VAE exhibited superior results not only in precision but also in other measures such as recall, F-score, and accuracy. This marks a significant advancement in trace retrieval research, addressing a big concern related to the precision of trace retrieval in recovering links between requirements and source code.

In future works, the authors will explore more intelligent solutions, based on unsupervised deep learning models, that can be applied to the same datasets (i.e., eTour, SMOS, and eANCI) to enhance the trace retrieval precision.

Conflicts of interest

The authors declare no conflict of interest

Authors contributions

Conceptualization, N. H. Al-walidi and N. R. Darwish; methodology, N. H. Al-walidi; validation, N. H. Al-walidi ; formal analysis, N. H. Al-walidi ; investigation, N. H. Al-walidi ; resources, N. H. Al-walidi and N. R. Darwish; data curation, N. H. Al-walidi and N. R. Darwish; writing—original draft preparation, N. H. Al-walidi; writing—review and editing, N. H. Al-walidi; visualization, N. H. Al-walidi; supervision, N. R. Darwish; project administration, N. R. Darwish.

References

[1] T. Du, B. Shen, G. Huang, Z. Yu, and D. Wu, “Automatic traceability link recovery via active learning”, *Frontiers of Information Technology & Electronic Engineering*, Vol. 21, No. 8, pp. 1217-1225, 2020.

[2] F. Wang, Z. Yang, B. Huang, Z. Liu, C. Zhou, Y. Bodeveix, and M. Filali, “An approach to generate the traceability between restricted natural language requirements and AADL models”, *IEEE Transactions on Reliability*, Vol. 69, No. 1, pp. 154-173, 2019.

[3] J. Zhu, G. Xiao, Z. Zheng, and Y. Sui, “Enhancing Traceability Link Recovery with Unlabeled Data”, In: *Proc. of IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE) Charlotte*, NC, USA, No. 33, pp. 446-457, 2022.

[4] C. Mills, J. E. Avila, A. Bhattacharya, G. Kondyukov, S. Chakraborty, and S. Haiduc, “Tracing with less data: active learning for classification-based traceability link recovery”, In: *Proc. of IEEE International Conf. On Software Maintenance and Evolution (ICSME) Cleveland*, OH, USA. pp. 103-113, 2019.

[5] L. Chen, D. Wang, J. Wang, and Q. Wang, “Enhancing Unsupervised Requirements Traceability with Sequential Semantics”, In: *Proc. of 26th Asia-Pacific Software Engineering Conference (APSEC)*, Putrajaya, Malaysia, No. 26, pp. 23-30, 2019.

[6] N. H. Alwalidi, S. Azab, A. Khamis, and N. Darwish, “Clustering-based Automated Requirement Trace Retrieval”, *International Journal of Advanced Computer Science and Applications*, Vol. 13, No. 12, pp. 783-792, 2022.

[7] Y. Liu, J. Lin, Q. Zeng, M. Jiang, and J. C. Huang, “Towards semantically guided

traceability", In: *Proc. of 2020 IEEE 28th International Requirements Engineering Conference (RE) Zurich, Switzerland*, No. 28, pp. 328-333, 2020.

[8] S. Wang, J. Cai, Q. Lin, and W. Guo, "An Overview of Unsupervised Deep Feature Representation for Text Categorization", *IEEE Transactions on Computational Social Systems*, Vol. 6, No. 3, pp. 504-517, 2019.

[9] M, Azar and L. Hamey, "Text summarization using unsupervised deep learning", *Expert Systems with Applications*, Vol. 68, No. 20, pp. 93-105, 2017.

[10] X. Zou, R. Settimi, and J. C. Huang, "Improving automated requirements trace retrieval: a study of term-based enhancement methods", *Empirical Software Engineering*, Vol. 15, No. 2, pp. 119-146, 2020.

[11] J. Guo, M. Gibiec, and J. C. Huang, "Tackling the term-mismatch problem in automated trace retrieval", *Empirical Software Engineering*, Vol. 22, No. 3, pp. 1103-1142, 2016.

[12] D. Qian and W. Cheung, "Enhancing variational autoencoders with mutual information neural estimation for text generation", In: *Proc: of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China, Vol. 1, No. 11, Vol. 9, pp. 4047-4057, 2019.

[13] R. Li, X. Li, G. Chen, and C. Lin, "Improving variational autoencoder for text modelling with timestep-wise regularisation", *arXiv Preprint arXiv*, Vol. 8, No. 13, pp. 2381-2397, 2020.

[14] R. Li, X. Li, C. Lin, M. Collinson, and R. Mao, "A stable variational autoencoder for text modelling", *arXiv Preprint arXiv:1911.05343*, Vol. 1, No. 11, pp. 1-6, 2019.

[15] L. Che, X. Yang, and L. Wang, "Text feature extraction based on stacked variational autoencoder", *Microprocessors and Microsystems*, Vol. 76, No. 7, pp. 1-10, 2020.

[16] K. Moran, D. Palacio, C. B. Cárdenas, D. McCrystal, D. Poshyvanyk, C. Shenefiel, and J. Johnson, "Improving the effectiveness of traceability link recovery using hierarchical bayesian networks", In: *Proc. of the ACM/IEEE 42nd International Conference on Software Engineering*, New York, USA, Vol. 20, No. 10, pp. 873-885, 2020.

[17] I. Lizarralde, C. Mateos, A. Zunino, T. Majchrzak, and T. Grønli, "Discovering web services in social web service repositories using deep variational autoencoders", *Information Processing & Management*, Vol. 57, No. 4, pp. 1-19, 2020.

[18] X. Fang, H. Bai, J. Li, Z. Xu, M. Lyu, and I. King, "Discrete auto-regressive variational attention models for text modelling", In: *Proc. of International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China, Vol. 123, No. 7, pp. 18-22, 2021.

[19] D. Cuddeback, A. Dekhtyar, and J. Hayes, "Automated requirements traceability: The study of human analysts", In: *Proc. of 18th IEEE International Requirements Engineering Conference, Sydney*, NSW, Australia, Vol. 18, No. 9, pp. 231-240, 2010.

[20] R. Lapeña, J. Font, C. Cetina, and Ó. Pastor, "Exploring new directions in traceability link recovery in models: The process models case", In: *Proc. of Advanced Information Systems Engineering: 30th International Conference, CAiSE 2018, Tallinn, Estonia, Proceedings. International Publishing*, Vol. 30, No. 5, pp. 356-373, 2018.

[21] W. Zogaan, "Towards an Intelligent System for Software Traceability Datasets Generation", *Rochester Institute of Technology*, Vol. 12 No. 17, pp. 1-132, 2019.

[22] A. M. Munoz and O. Rendon, "An approach based on fog computing for providing reliability in IoT data collection: A case study in a Colombian coffee smart farm", *Applied Sciences*, Vol. 10, No. 24, pp. 1-16, 2020.

[23] X. Zou, R. Settimi, and J. C. Huang, "Improving automated requirements trace retrieval: a study of term-based enhancement methods", *Empirical Software Engineering*, Vol. 15, No. 2, pp. 119-146, 2010.

[24] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. C. Huang, "Traceability transformed: Generating more accurate links with pre-trained Bert models", In: *Proc. of IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, Madrid, ES, Vol. 43, No. 5, pp. 324-335, 2021.

[25] K. Moran, D. Palacio, N. B. Cárdenas, C. McCrystal, D. Poshyvanyk, D. Shenefiel, and J. Johnson, "Improving the effectiveness of traceability link recovery using hierarchical bayesian networks", In: *Proc. of the ACM/IEEE 42nd International Conference on Software Engineering*, Vol. 42, No. 10, pp. 873-885, 2020.

[26] P. Sharma, "Datasets Used in Fifteen Years of Automated Requirements Traceability Research", *Rochester Institute of Technology*, No. 12, pp. 1-90, 2017.

[27] T. Li, S. Wang, D. Lillis, and Z. Yang, "Combining machine learning and logical reasoning to improve requirements traceability recovery", *Applied Sciences*, Vol. 10, No. 20, pp. 1-23, 2020.

[28] CoEST: Center of excellence for software traceability, http://www.CoEST.org-2023.

[29] C. Mills, J. E. Avila, A. Bhattacharya, G. Kondyukov, Chakraborty, and S. Haiduc, "Tracing with less data: active learning for classification-based traceability link recovery", In: *Proc. of IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Cleveland, OH, USA, Vol. 4, No. 9, pp. 103-113, 2019.

[30] F. Faiz, R. Easmin, and A. Gias, "Achieving Better Requirements to Code Traceability: Which Refactoring Should Be Done First?", In: *Proc. of 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, Lisbon, Portugal, Vol. 10, No. 9, pp. 9-14, 2016.

[31] B. Lund and T. Wang, "Chatting about ChatGPT: how may AI and GPT impact academia and libraries?", *Library Hi Tech News*, Vol. 40, No. 3, pp. 26-29, 2023.

[32] W. Jiao, W. Wang, J. Huang, X. Wang, and Z. Tu, "Is ChatGPT a good translator? A preliminary study", *arXiv preprint arXiv:2301.08745*, Vol. 1, No. 1, pp. 1-8, 2023.

[33] O. Toporkov and R. Agerri, "On the Role of Morphological Information for Contextual Lemmatization", *arXiv Preprint arXiv:2302.00407*, Vol. 1, No. 2, pp. 1-30, 2023.

[34] S. Akuma and P. Anendah, "New Query Expansion Approach for Improving Web Search Ranking", *Information Technology and Computer Science*, Vol. 15, No. 1, pp. 42-55, 2023.

[35] M. Maziarz, L. Grabowski, T. Piotrowski, E. Rudnicka, and M. Piasecki, "Lexicalized and Non-lexicalized Multi-word Expressions in WordNet: a Cross-encoder Approach", *hitz.eus*, pp.1-7, 2023.

[36] M. Khatun, "Evaluating Word Embedding Models for Traceability", *Doctoral dissertation, Louisiana State University and Agricultural and Mechanical College*, Vol. 5414, No. 7, pp. 1-80, 2021.

[37] R. Marco and S. Ahmad, "Conditional Variational Autoencoder with Inverse Normalization Transformation on Synthetic Data Augmentation in Software Effort Estimation", *International Journal of Intelligent Engineering and System*, Vol. 15, No. 3, pp. 366-381, 2022, doi: 10.22266/ijies2022.0630.31.

[38] A. Maulana, C. Fatichah, and N. Suciati, "Facial Inpainting Using Generative Adversarial Network with Feature Reconstruction and Landmark Loss to Preserve Spatial Consistency in Unaligned Face Images", *International Journal of Intelligent Engineering and System*, Vol. 13, No. 6, pp. 219-228, 2020, doi: 10.22266/ijies2020.1231.20.

[39] O. Boyar and I. Takeuchi, "Latent Reconstruction-Aware Variational Autoencoder", *arXiv Preprint arXiv:2302.02399*, Vol. 3, No. 2, pp. 1-20, 2023.

[40] K. Han, H. Wen, J. Shi, J. Lu, K. Zhang, D. Fu, and Z. Liu, "Variational autoencoder: An unsupervised model for encoding and decoding fMRI activity in visual cortex", *Neuro Image*, Vol. 198, No. 9, pp. 125-136, 2019.

[41] K. Rungta, G. Chau, A. Dewangan, M. Wagner, and L. Huang, "Sentence Generation and Classification with Variational Autoencoder and BERT", *margotwagner.com*, No.2, pp.1-11, 2022.

[42] R. Wei, C. Garcia, A. E. Sayed, V. Peterson, and A. Mahmood, "Variations in variational autoencoders-a comparative evaluation", *IEEE Access*, Vol. 8, No. 8, pp. 153651-153670, 2020.

[43] A. Asperti and M. Trentin, "Balancing reconstruction error and kullback-leibler divergence in variational autoencoders", *IEEE Access*, Vol. 8, No. 11, pp. 199440-199448, 2020.

[44] https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/, 2023.

[45] Y. Qiu and X. Wang, "Stochastic approximate gradient descent via the Langevin algorithm", In: *Proc. of the AAAI Conference on Artificial Intelligence*, New York, USA, Vol. 34, No. 04, pp. 5428-5435, 2020.

[46] A. Sharma, K. Aggarwal, G. Bhardwaj, S. Chakrabarti, P. Chakrabarti, T. J. Abawajy, and H. Mahdin, "Classification of Indian classical music with time-series matching deep learning approach", *IEEE Access*, Vol. 9, No., pp. 102041-102052, 2021.

[47] T. Du, B. Shen, G. Huang, Z. Yu, and D. Wu, "Automatic traceability link recovery via active learning. Frontiers of Information Technology and Electronic Engineering", *Frontiers of Information Technology & Electronic Engineering*, Vol. 21, No. 8, pp. 1217-1225, 2020.

[48] T. Li, S. Wang, D. Lillis, and Z. Yang,

"Combining machine learning and logical reasoning to improve requirements traceability recovery", *Applied Sciences*, Vol. 10, No. 20, pp. 1-23, 2020.