

# **Comparative Analysis of Word Embedding Techniques on Software Defect Prediction**

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

**MASTER OF TECHNOLOGY**

IN

**DATA SCIENCE**

Submitted by:

**GAURAV SHARMA**

**2K22/DSC/05**

Under the supervision of

**Miss. PRIYA SINGH**

**Assistant Professor**



**DEPARTMENT OF SOFTWARE ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

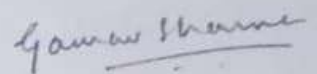
Bawana Road, Delhi – 110042 MAY, 2024

## ACKNOWLEDGEMENT

The achievement of Major Project II necessitates the assistance and support of a large number of individuals and an organisation. This project's report writing opportunity allows me to thank everyone who contributed to the project's successful completion. I would want to express my sincere gratitude to my supervisor, Miss Priya Singh, for allowing me to work on my project under her supervision. Thank you very much for your support, encouragement and suggestions; without then, our work would not have been successful. Her unwavering support and inspiration helped me to see that the process of learning is more important than the end result.

I want to express my sincere thanks to the faculty and personnel at the institution for providing us with a infrastructure, laboratories, library, suitable educational resources, testing facilities, and a working atmosphere that didn't interfere with our ability to complete our work.

I would also like to thank all of my friends and classmates for thir unwavering support. They have assisted me in every way, providing me with fresh ideas, the knowledge I needed, and the will to finish the assignment. I want to express my gratitude to my parents for always supporting me after finishing my task.



Gaurav Sharma

(2K22/DSC/05)

M.Tech (Data Science)

Delhi Technological University

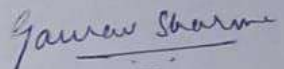
**DEPARTMENT OF SOFTWARE ENGINEERING DELHI  
TECHNOLOGICAL UNIVERSITY (Formerly Delhi College of  
Engineering) Bawana Road, Delhi-110042**

**CANDIDATE'S DECLARATION**

I, Gaurav Sharma, 2K22/DSC/05 student of M. Tech (DSC), hereby declare that the project entitled "Comparative Analysis of Word Embedding Techniques in Software Defect Prediction" which is submitted by me to Department of Software Engineering, Delhi Technological University, Shahbad Daultpur, Delhi in partial fulfilment of requirement for the award of the degree of Master of Technology in Data Science, has not been previously formed the basis for any fulfilment of requirement in any degree or other similar title or recognition. This report is an authentic record of my work carried out during my degree under the guidance of **Miss. Priya Singh**.

Place: Delhi

Date: 27<sup>th</sup> May 2024

  
Gaurav Sharma

(2K22/DSC/05)

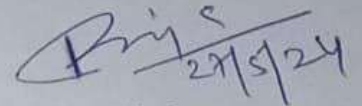
**DEPARTMENT OF SOFTWARE ENGINEERING DELHI  
TECHNOLOGICAL UNIVERSITY (Formerly Delhi College of  
Engineering) Bawana Road, Delhi-110042**

**CERTIFICATE**

I hereby certify that the project entitled "Comparative Analysis of Word Embedding Techniques in Software Defect Prediction" which is submitted by Gaurav Sharma (2K22/DSC/05) to Department of Software Engineering, Delhi Technological University, Shahbad Daulatpur, Delhi in partial fulfilment of requirement for the award of the degree of Master of Technology in Data Science, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this university or elsewhere.

Place: Delhi

Date: 27/5/24



Miss. Priya Singh

(Assistant Professor, SE, DTU)

## ABSTRACT

Embeddings are known for their ability to understand semantic relationships, reduce dimensionality, and identify patterns in data. These techniques are mostly used in machine learning as they are helpful and can easily be integrated into prediction models. Embedding techniques such as Word2Vec, TF-IDF, FastText, and Doc2Vec are commonly used for software defect prediction tasks. While creating a defect prediction model, picking the suitable embedding method is very important. This study undertakes a comprehensive comparison of these widely-used embedding techniques within the realm of software defect prediction. The analysis is based on a diverse set of Java projects sourced from the open-source Promise repository. The evaluation process involved training and testing multiple deep learning models to assess the effectiveness of each embedding technique. Several key evaluation metrics, including the Matthews correlation coefficient (MCC), specificity accuracy, precision, recall, and F1 score, were used to measure performance. The results of this rigorous evaluation reveal that Doc2Vec significantly outperforms the other embedding techniques, demonstrating its superiority in capturing semantic nuances and contributing to more accurate defect predictions. FastText emerges as the second-best performer, surpassing TF-IDF and Word2Vec in various metrics. TF-IDF, while effective, falls short of the performance levels achieved by Doc2Vec and FastText, but still surpasses Word2Vec, which ranks last in this comparison.

# TABLE OF CONTENT

	Page No.
<b>ACKNOWLEDGEMENT</b>	<b>ii</b>
<b>CANDIDATE’S DECLARATION</b>	<b>iii</b>
<b>CERTIFICATE</b>	<b>iv</b>
<b>ABSTRACT</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>ix</b>
<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
<b>CHAPTER 1: INTRODUCTION</b>	<b>11</b>
1.1 Motivation	11
1.2 Problem Statement	11
1.3 Objective	11
1.4 Dissertation Organisation	12
<b>CHAPTER 2: BACKGROUND</b>	<b>13</b>
2.1 Software Defect Prediction	13
2.2 SDP Using Deep Learning	13
2.3 Artificial Neural Network	14
2.4 Convolutional Neural Network	14
2.5 Recurrent Neural Network	15
2.5.1 LSTM	16
2.5.2 GRU	16
2.6 Embedding Techniques	16
2.7 Word2Vec	17

2.8 TF-IDF	18
2.9 FastText	19
2.10 Doc2Vec	19
<b>CHAPTER 3: LITERATURE REVIEW</b>	<b>20</b>
<b>CHAPTER 4: METHODOLOGY</b>	<b>22</b>
4.1 Corpus Generation Using AST	22
4.2 Generation of Sequence Tokens	22
4.3 Fine Tuning of Pre-trained models	22
4.4 Generation of Embeddings	23
4.5 Comparison of Techniques	23
<b>CHAPTER 5: EXPERIMENTAL SETUP</b>	<b>24</b>
5.1 Dataset Used	24
5.2 Evaluation Metrics	24
5.3 Hyperparameter Settings	25
<b>CHAPTER 6: RESULT</b>	<b>26</b>
<b>CHAPTER 7: CONCLUSION AND FUTURE WOEK</b>	<b>33</b>
<b>REFERENCES</b>	<b>34</b>
<b>LIST OF PUBLICATION</b>	<b>36</b>

## LIST OF FIGURES

		<b>Page No.</b>
Figure 2.1	Flowchart of Software Defect Prediction	13
Figure 2.2	Artificial Neural Network	14
Figure 2.3	Convolution Neural Network	15
Figure 2.4	Recurrent Neural Network	15
Figure 2.5	Architectural Design of GRU and LSTM	16
Figure 2.6	Illustration of Skip-NGram and CBOW model	17
Figure 2.7	Binary tree structure followed by FastText	19
Figure 2.8	Distributed memory architecture for Doc2Vec model	20
Figure 4.1	Process of Defect Prediction	24
Figure 6.1	Average values of FNR and FPR	33



## LIST OF TABLES

	<b>Page No.</b>
Table 5.1 Description of project along with their version	25
Table 6.1 CNN based model using Word2Vec Embeddings	27
Table 6.2 CNN based model using TF-IDF Embeddings	28
Table 6.3 CNN based model using FastText Embeddings	28
Table 6.4 CNN based model using Doc2Vec Embeddings	28
Table 6.5 ANN based model using Word2Vec Embeddings	29
Table 6.6 ANN based model using TF-IDF Embeddings	29
Table 6.7 ANN based model using FastText Embeddings	29
Table 6.8 ANN based model using Doc2Vec Embeddings	30
Table 6.9 GRU based model using Word2Vec Embeddings	30
Table 6.10 GRU based model using TF-IDF Embeddings	30
Table 6.11 GRU based model using FastText Embeddings	31
Table 6.12 GRU based model using Doc2Vec Embeddings	31
Table 6.13 LSTM based model using Word2Vec Embeddings	31
Table 6.14 LSTM based model using TF-IDF Embeddings	32
Table 6.15 LSTM based model using FastText Embeddings	32
Table 6.16 LSTM based model using Doc2Vec Embeddings	32
Table 6.17 Average performance with Word2Vec and TF-IDF embeddings	33
Table 6.18 Average performance with FastText and Doc2Vec embeddings	33

## **LIST OF ABBREVIATIONS**

SDP: Software Defect Prediction

CNN: Convolution Neural Network

ANN: Artificial Neural Network

RNN: Recurrent Neural Network

NLP: Natural Language Processing

CBOW: Continuous Bag of Words

LSTM: Long Short Term Memory

GRU: Gated Recurrent Unit

MCC: Matthews Correlation Coefficient

TF-IDF: Term Frequency

Word2Vec: Word to Vector

Doc2Vec: Document to Vector

FNR: False Negative Rate

FPR: False Positive Rate

# CHAPTER 1

## INTRODUCTION

### 1.1 Motivation

Software development is inherently complex and fraught with numerous challenges. Lots of companies are now wasting colossal amounts of their money and resources fixing bugs within the computer software. The bugs might be small, but really set up big issues in terms of quality and how the system operates. If such bugs are not fixed, then faulty systems will grind to an embarrassing stop. This has an effect on user experience and stability of the system, from a consumer's perspective. The software defect prediction (SDP) technique is a powerful tool to tackle this situation. It helps avoid such faults much before they could be visible in terms of consequences. Such an approach would be very important if excellence in software engineering is to be achieved. In this view, it becomes indispensable to enrich the efficacy of SDP for expediting the development process of useful software.

### 1.2 Problem Statement

SDP is an AI software tool used by developers in dealing with the problems related to their software. It works in hand with predictive models, which focus more on innovation. In fact, all such innovations come through the techniques associated with NLP, ML, and DL. These include different types of embeddings, such as Word2Vec, GloVe, FastText, TF-IDF, and Doc2Vec. These techniques help to convert words into numerical vectors for the algorithms so that they are able to pick up the semantic relationships present in textual data.

Although these have shown promise in improving SDP, there still lacks comprehensive comparison across different word embedding techniques within the context of SDP. Thus, there is a need to understand the relative performance of the different embedding techniques employed over models for SDP tasks. The necessity to confirm that diverse SDP models, integrating embedding techniques, will align seamlessly with defect prediction models.

### 1.3 Objective

This thesis aims to contribute significantly to the field of SDP by examining the impact of various embedding techniques on SDP tasks. Specifically, it seeks to achieve the following objectives:

1. Examine the impact of various embedding techniques on SDP tasks.
2. Conduct a comparative evaluation of commonly used embedding methods, such as Word2Vec, TF-IDF, FastText and Doc2Vec.
3. Analyze the performance of different embedding techniques across multiple evaluation metrics specific to SDP.
4. Evaluate how effective the DL models are when employed for SDP.
5. Investigate how the integration of embedding techniques with DL models enhances SDP outcomes.
6. Provide insights into selecting suitable embedding techniques tailored to specific SDP requirements.

Through these objectives, the study aims to tell how effective embedding techniques are in enhancing SDP and offer guidance for researchers in selecting the most suitable techniques for their SDP tasks.

## 1.4 Dissertation Organisation

The thesis is structured to systematically and comprehensively address the objectives. Chapter 1 consists of Introduction which provides the overview of the project clarifying the main objectives and discusses the motivation behind the thesis along with the Problem Statement. Chapter 2, deals with the Background wherein the background of the research work is dependent. It generally deals with the tech stacks and the important components that is essential and used majorly in the thesis. Chapter 3 provides us with the Literature Review or the Related work done in the domain of SDP. It summarises and gives the overview of existing research on the application of sequential models for software defect prediction. It covers methodologies, datasets, evaluation metrics, and case studies. In Chapter 4 goes with the detailed methodology including search strategies. It gives the detailed description of all the steps that is followed in the experiment with the inclusion criteria. It also covers the data extraction methods, corpus generation methods, how the model is pre-trained and what models have been considered for creating the embeddings along with the DL models that were used to actually predict the defects in software. It also presents us with the dataset discussion along with every version considered for the experiment. Chapter 5, consists of Experimental Setup that is every setup is defined in this section. For elaboration this chapter contains the evaluation metrics which helps to derive the conclusion of which embedding methods are suitable for Defect prediction model used in the research. This chapter also discusses about the hyperparameter settings which were set in order to get the desired result. Then finally comes the Chapter 6. That is Result and Discussion Section where tables and figure are provided giving defining the result that how things were set up and finally the result is achieved. In this section the technical details are present with examples of how each and everything is mentioned and taken into consideration. Chapter 7 is the Conclusion section that concludes the result. Along with it tells about the future research in the domain of SDP . Lastly the Reference section Chapter 8 is provided that shares the materials from where the inspiration had been taken.

## CHAPTER 2

### BACKGROUND

#### 2.1 Predicting Software Defects

SDP is a process of using ML and DL techniques to predict the likelihood of software defects in a software system. The goal of SDP is to identify potential defects before they occur, enabling software development teams to take preventive measures and improve software quality.

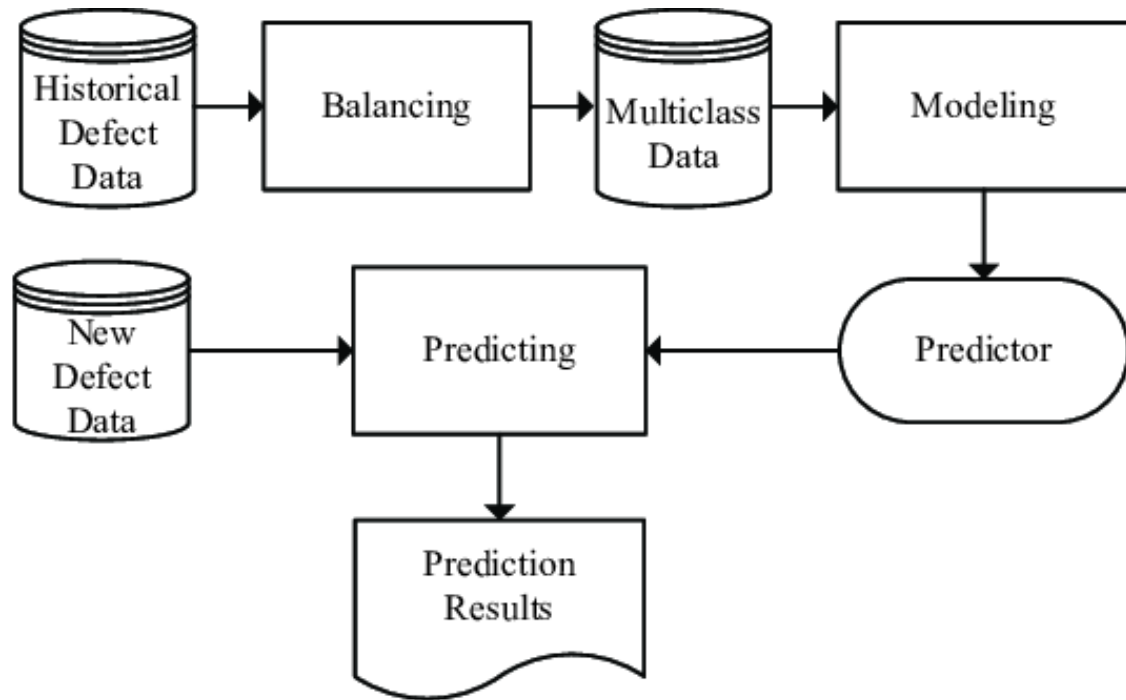


Figure 2.1 Flowchart of software defect prediction

It can be challenging due to the dynamic nature of software development, where software code and requirements can change rapidly over time. Additionally, it can be difficult to obtain accurate historical data on defects and software metrics, especially in cases where the software system is new or has not yet been deployed.

#### 2.2 SDP Using DL Methods

For years, ML techniques have played a crucial role in SDP tasks, offering valuable insights into potential defects through features extracted from software artifacts. However, traditional ML models face challenges when dealing with the complexity and intricacies of textual and image data. In contrast, DL techniques have proven to be highly effective for these data types. With their capability to automatically learn hierarchical representations from raw data, DL models excel at identifying the nuanced patterns and relationships within textual and image data. These methods encompass various neural network architectures. Some of the architecture such as Artificial Neural Networks, Convolution Neural Networks and Recurrent Neural Networks are often used for SDP.

## 2.3 Artificial Neural Networks

A unit is an artificial neuron and is considered the fundamental unit of an artificial neural network. In essence, the Artificial Neural Network in a system consists of several layers of units. A layer can have a few dozen units or possess millions of them, all depending upon how complex neural networks are to be discovered for the data set. A typical artificial neural network is composed of input layers, output layers, and hidden layers. The input layer is where data comes in from the rest of the world, which the neural network is to appraise or learn about. This data will then pass through one or more hidden layers to convert the input into a more useful form for the following layer: the output layer. Finally, the output layer gives an output response to the ANNs of the input data given.

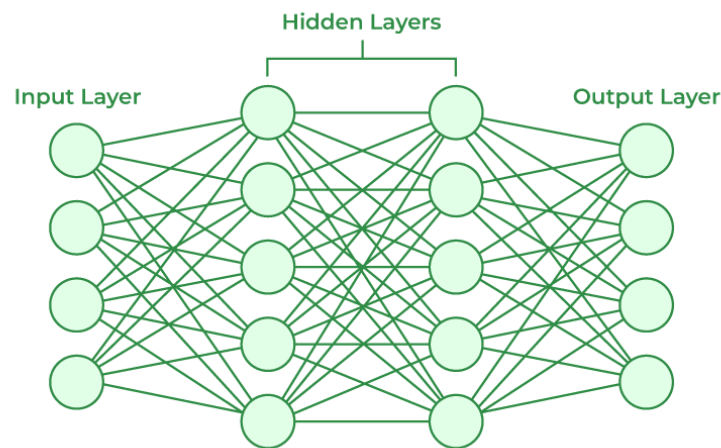


Figure 2.2 Artificial Neural Network

It is within this respect that ANNs are rather critical in the detection of bugs, since complex relationships between software data can be well modeled. With these ANNs being layered, they automatically identify useful features out of large datasets representing code metrics, bug reports, and textual data. This makes it possible for ANNs to identify probable defective patterns, allowing clear management proactively by means of the early detection of software problems. ANNs improve the accurateness and relevant SDP models by using their deep learning capabilities, making it better quality software. They are also making it more dependable in the life cycle of development. They are efficient in contemporary designed software development processes due to the flexibility of data types.

## 2.4 Convolution Neural Networks

CNNs are an enhanced form of artificial neural networks, primarily utilized for feature extraction from grid-like matrix datasets. Take visual datasets, such as pictures or videos, where data patterns are very important. The input layer, pooling layer, convolutional layer, and fully connected layers are some of the layers that make up a convolutional neural network. The architectural design can be seen in figure 2.3.

The input is processed by the convolutional layer to extract features, the pooling layer reduces computation by downsampling the data, and the fully connected layer generates the final prediction. By using gradient descent and backpropagation, the network discovers the best filters. CNN are generally preferred for image data as they excel in feature extraction. However, 1-D CNN also performs well in textual data and hence it is used in areas such as NLP.

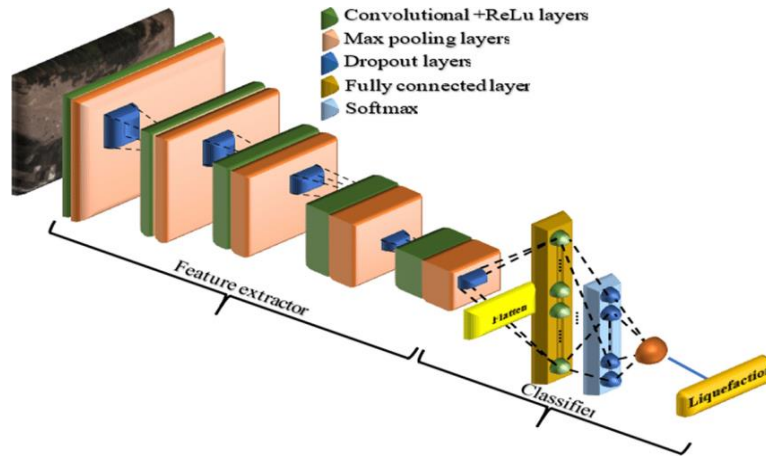


Figure 2.3 Convolution Neural Network

CNNs significantly enhance SDP by effectively processing and analyzing code and textual data. CNNs are adept at identifying spatial hierarchies and patterns within data through their convolutional layers, making them particularly useful for examining code snippets, source code images, and structured text. By automatically learning and extracting features relevant to defect prediction, CNNs improve the accuracy and efficiency of detecting potential software bugs. Their ability to handle large and complex datasets ensures robust SDP models, ultimately contributing to higher software quality and more reliable development processes.

## 2.5 Recurrent Neural Networks

A type of neural network called a RNN uses the output from the preceding step as the input for the current step. In classical neural networks, all the inputs and outputs are independent of each other. However, there occur situations where, due to the previous words, a guess has to be made regarding what word is in the next sentence, therefore, they must be remembered. So, to tackle that problem an RNN with the help of a Hidden Layer was formed. The Hidden state is the most essential feature of an RNN; it's mostly the same with the sequence history. Due to the fact that an RNN retains memory associated with the previous input into the network, it's also referred to as the Memory State.

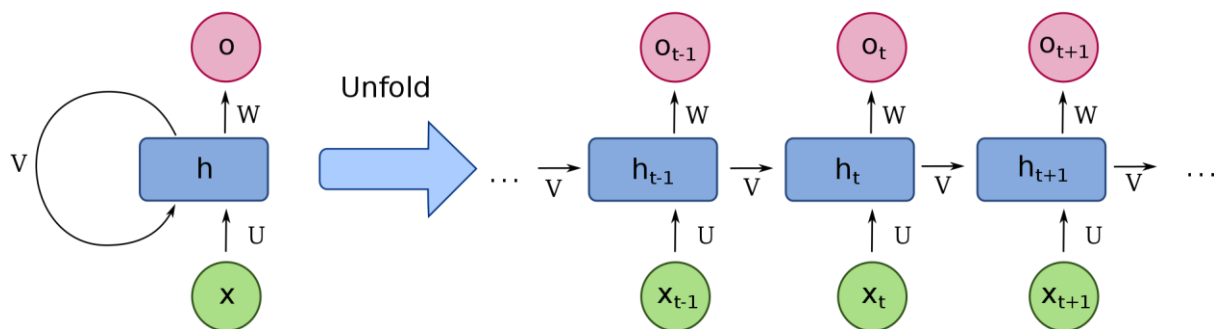


Figure 2.4 Recurrent Neural Network

RNNs are ideally suited to be used with SDP because they can deal with sequential data and capture dependencies over time. These architectures are more than well known for tasks in which the order of events and contexts really matters, such as changes to your code, history of commits, and bug report analysis through time. This is mainly because of the recurrence imputed in them, hence equipping RNNs with the intrinsic ability of tracking long-term dependencies by storage of a memory of past or

previous input sequences in ways of determining the dynamic understanding of how software grows and defects potentially introduced.

### 2.5.1 Long Short-Term Memory Networks

The RNNs of this special form, called LSTM networks, manage to alleviate the vanishing-path problem, widely present in traditional RNNs. LSTMs use a chain of input, forget, and output gates to control the flow of information in such a way that relevant information over a large time step can be maintained, while irrelevant data may be ignored. This is particularly valuable in SDP, where very long sequences of changes or histories of extended bugs lead to early changes that significantly affect later outcomes. LSTMs increase the predictive accuracy of SDP models by capturing long-term dependencies and allow them to identify complex patterns of defects that span very long software development periods.

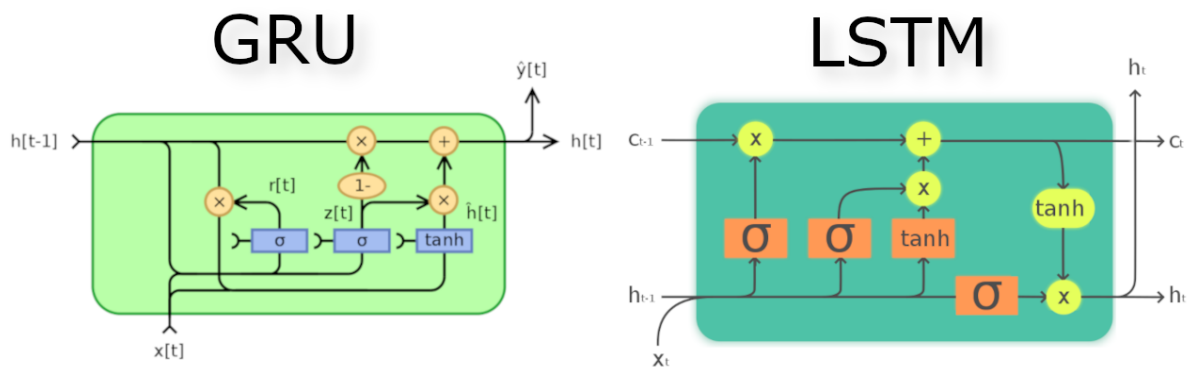


Figure 2.5 Architectural Design of GRU and LSTM

### 2.5.2 Gated Recurrent Units (GRU)

In fact, another variant of the RNN was developed to keep its good properties of being able to deal with long-term dependencies while at the same time reducing the complexity: the so-called Gated Recurrent Unit. In fact, GRUs fold together the input and forget gates, and cell and hidden states, into one update gate. This is computationally less complex of a computation during back prop and computationally faster to run than LSTMs because it has fewer gates, yet it still copes quite well with sequential dependencies. Hence, GRUs can settle on a reasonable balance between performance and efficiency, depending on the needs of real-time defect prediction in the codebase evolution analysis with SDP. Being lightweight in computation, GRU is competent enough to fit into this task; therefore, it can be a good choice for installing defect prediction within pipelines for CI and CD.

## 2.6 Embedding Techniques

High-dimensional data can be represented in various embedding techniques which internalize semantic relationships into dense, low-dimensional vectors. Embeddings represent words, phrases, or even entire documents as vectors within a continuous vector space based on both syntactic relationships and contextual meaning. Word embeddings are done with respect to approaches that look at large corpora like Word2Vec, GloVe, and FastText, enabling useful and meaningful comparisons. Embeddings are important to both ML and DL tasks because they provide a way to transform complex data into forms that can be handled by the model in use, thus helping sentiment analysis, text classification, and other tasks.



Heterogeneous and complex software data are made meaningful and processable by embedding techniques in the context of SDP tasks. The truly complex patterns living within a software artifact might not have been fully captured by the traditional approaches followed in SDP, which mostly relied on manually created features such as code metrics and historical defect data. The methodologies, including Word2Vec, Doc2Vec, TF-IDF, and FastText embeddings, encode syntactic and semantic information extracted from the source code and its surrounding documentation into continuous vector spaces. These embeddings have realized scalability and adaptability of SDP models, allowing for transfer learning where the models trained over one project are applied over successfully to others.

In addition, embedding is expected to improve the handling of large codebases and diversified data sources, such as bug reports, commit messages, and documentation, by automatizing the process of feature extraction, thus reducing the need for human intervention. This way, resulting SDP models will provide much more fine-grained representation of software artifacts, making them on a much higher level of accuracy and prediction for more reliable and efficient processes of software development.

## 2.7 Word2Vec

A very common word embedding technique for word2vec in NLP aids in converting words into blasé vectors of a very low dimension and retains the semantics of the word. Word2Vec is a creation by Google, and it learns distributed representations of words based on their contextual usage across a corpus. The model has two primary architectures: skip-gram and continuous bag of words (CBOW).

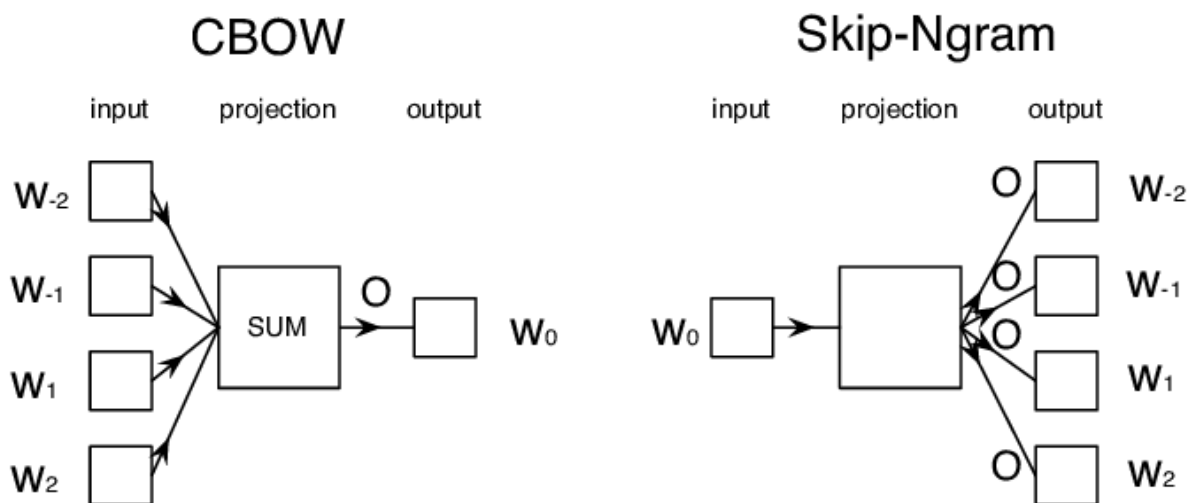


Figure 2.6 Illustration of Skip-NGram and CBOW model

The architecture of the CBOW model predicts a target word from its surrounding context words. For example, if we take the sentence "The cat sat on the mat" and we take the word "cat" as the target predictive word, then the words "The," "sat," "on," and "the" will be considered context words in CBOW for predicting "cat." The vector representations of these context words are needed to be summed up, or in other words aggregated, for an architecture to predict the target word. It works well in practice for small training sets and gives good results generally when the words are indeed very common.

Given the target word, the Skip-gram architecture predicts the words before it. For example, "The," "sat," "on," and "the" are the context words that Skip-gram looks to predict the target word "cat." Finally, in order to train, Skip-gram forms context pairs by treating each word as an individual

observation. Skip-gram needs a high amount of training data. In contrast, it did better on larger datasets and much more competently with less frequent words.

## 2.8 TF-IDF

Term frequency-inverse document frequency, or TF-IDF for short, is a statistical metric that is frequently used in text mining and information retrieval to assess a word's significance in relation to a set of documents. Inverse Document Frequency and Term Frequency are two distinct concepts. Together, they make up TF-IDF.

A term's frequency of occurrence in a document is assessed using the Term Frequency (TF) measure. Because every document is different in length, a term may appear far more frequently in longer documents than in shorter ones.

$$TF(t) = \frac{TN(\text{Number of times term } t \text{ appears in a document})}{\text{Total number of terms in the document}}$$

On the other hand, the Inverse Document Frequency (IDF) component, which is derived from the logarithm of the ratio of the total number of documents to the number of documents containing the term, determines how uncommon a term is throughout the corpus. Terms that are uncommon throughout the corpus but frequently occur in a document have high TF-IDF scores, indicating their importance in characterizing the content of the document. The following is the formula for term frequency.

$$IDF(t) = \frac{\log_e(\text{Total number of documents})}{\text{Number of documents with term } t \text{ in it}}$$

The TF and IDF scores for each term in a document are multiplied to create TF-IDF scores, which are weighted scores that represent a term's importance both locally within the document and globally throughout the corpus. High TF-IDF score words are valued and frequently utilized as features or keywords in a variety of text analysis tasks, such as keyword extraction, text classification, and document ranking. The flexible and popular TF-IDF approach helps to extract valuable insights and patterns from textual data by giving textual information a numerical representation.

## 2.9 FastText

FastText is a state-of-the-art word embedding technique developed by Facebook's AI Research (FAIR) lab. It builds upon the success of Word2Vec but introduces several improvements, particularly in handling out-of-vocabulary words and subword information. FastText trains word vector models very quickly. In less than ten minutes, you can train roughly one billion words.

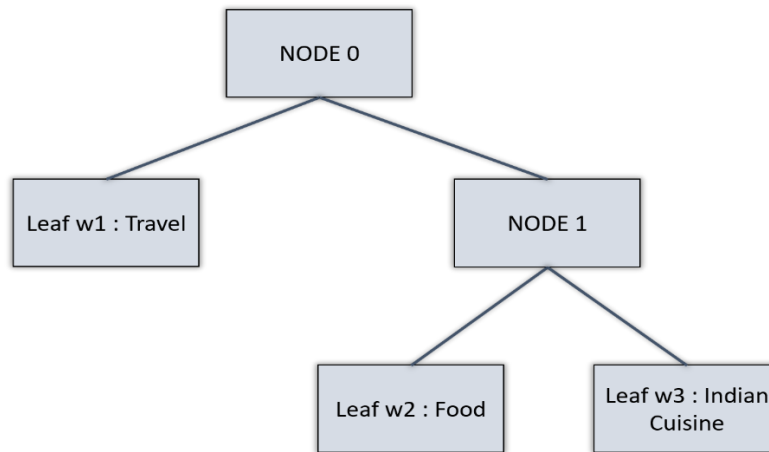


Figure 2.7 Binary tree structure followed by FastText

It uses a binary tree to represent the labels in this method. In a binary tree, each node stands for a probability. The probability along the path leading to a given label represents that label. This indicates that the labels are represented by the binary tree's leaf nodes.

Deep neural network models can be difficult to train and test. These methods use a linear classifier to train the model. Using this model, one can develop algorithms to obtain vector representations of words through supervised or unsupervised learning. It also evaluates these models. FastText supports both the CBOW and Skip-gram models. FastText handles uncommon and out-of-vocabulary words efficiently by representing words as bags of character n-grams, in contrast to Word2Vec, which represents each word as a single vector. This enables it to capture morphological information. FastText is especially helpful for tasks involving large vocabularies or languages with complex morphology because it can generate embeddings for words that are not present in the training data by taking subword information into consideration.

Like Word2Vec, FastText trains on a skip-gram model with negative sampling; however, it incorporates character n-grams into the model architecture. This makes it possible for FastText to record more detailed semantic information and enhance the quality of word embeddings, particularly in tasks involving uncommon or misspelled words and morphologically rich languages.

## 2.10 Doc2Vec

Extending the Word2Vec model, Doc2Vec is a neural network-based method for producing document embeddings. Documents can be represented as continuous vector representations thanks to Doc2Vec, which learns embeddings for entire documents in contrast to Word2Vec, which learns embeddings for individual words.

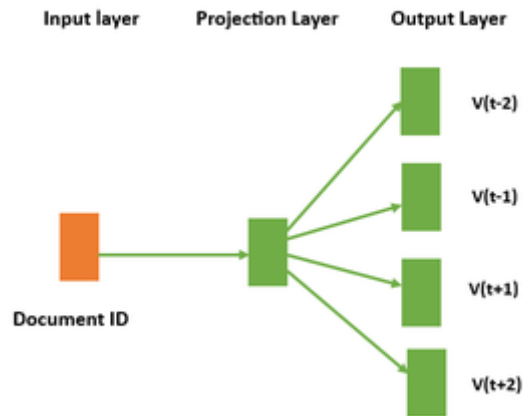


Figure 2.8 Distributed memory architecture for Doc2Vec model

The Paragraph Vector model, which comes in two flavors: Distributed Memory (DM) and Distributed Bag of Words (DBOW), is the most widely used implementation of Doc2Vec. With the help of document vector and surrounding word vectors, DM variant model gets to predict the next word in a context. On the other hand, DBOW uses only the document vector to guess randomly chosen words from the text. For example to create dense fixed-length vectors that can capture semantic meaning and contexts of a document, Doc2Vec uses neural networks to extract semantic information from documents. These embeddings are useful for document classification, grouping and similarity retrieval tasks in textual analysis where relationships between documents are maintained.

## CHAPTER 3

### LITERATURE REVIEW

Word embeddings work by representing text in  $n$ -dimensional space. They are essential for solving NLP-related problems. One such problem in identifying Swahili Smishing communications directed at mobile money customers is emphasized by S. Iddi et al. [3]. These techniques allow for efficient classification by capturing semantic links in text, which is crucial for differentiating genuine messages from smishing ones. Similarly, to establish a unified feature space for text and image modalities, Zongwei [4] introduced a multi-modal approach by integrating TF-IDF features with LSTM networks for capturing sequential information. The incorporation of TF-IDF helps in refining the image modalities.

Emotion processing is becoming an important research area in fields such as data analysis and NLP. For analyzing emotion, it is important to capture the presence of specific words along with their relationships with other words. Sabery [5] proposed a hybrid model for emotion analysis by combining the Deep Belief Network with TF-IDF and Glove. The embeddings helped outperform the baseline models in several metrics. In a similar context of emotion analysis, Canales [6] achieved efficient data annotation through Word2Vec embeddings, enhancing the categorization process of different emotions.

In the context of SDP, the role of defect prediction models becomes equally important as the embedding techniques. For a longer period of time, conventional ML approaches have been used in creating prediction models. The issue with these approaches lies in capturing the semantic relationship among textual data. In comparison to these approaches, neural networks perform better with image and textual data. Using a DL approach, Miholca [7] significantly improved defect prediction, outperforming conventional methods in the Calcite program. The significance of SDP is also highlighted by Nevendra [8] in concerns regarding software complexity. The research shows notable performance gains by comparing DL techniques across open-source projects. This change in strategy creates new opportunities for improving defect prediction models.

Using hybrid features [9] is advantageous, making defect prediction models more flexible. Wang cleverly combined the AST and Control Flow Graph (CFG) via the Graph Isomorphism Network to push SDP with H-GIN as evidence. With respect to the PyTraceBus dataset, H-GIN demonstrated better prediction accuracy than earlier approaches. Similarly, graph neural networks (GNN) and transformers were used to create a novel model that Tang and He presented [10]. Their technique, which included absolute and relative locations in the AST, addressed the local learning constraints of GNN and showed superior F-measure and improved detection of faulty features on the PROMISE dataset.

Another hybrid model Siamese dense neural networks (SDNNs) proposed by Zhao et al. [R16], which capitalize on their capacity to learn from small amounts of data. To improve prediction accuracy, SDNNs combine learning of distance metrics and similarity features. In order to capture high-level similarity features and use a contrast loss function for prediction, the model is constructed and trained in two stages. The competitive performance of SDNNs is demonstrated through comparison with state-of-the-art SDP approaches across 10 datasets, exhibiting notable improvements in prediction accuracy when compared to benchmarked methods.

A dataset of more than 400 thousand articles from design pattern books was used by D. Liu et al. for DPWord2Vec [11], a technique to concurrently embed design patterns and natural language words

into vectors. According to evaluation, DPWord2Vec performs 24.2\%–120.9\% better than baseline algorithms when assessing word and design pattern similarity. Additionally, DPWord2Vec enhances design pattern tasks by 6.5\%–70.7\%, including tag suggestion and selection. A similar approach to learning from datasets was used in a technique for proposing Web services for superior Mashup applications put forth by B. Cao et al. [12]. Their method uses Word2Vec for semantic representations from service descriptions and creating a service relationship network, combining bilinear graph attention representation with xDeepFM quality prediction. The findings on the ProgrammableWeb dataset demonstrate better performance in terms of accuracy and recall compared to other approaches.

The effect of issue classification using data from seven open-source repositories on SDP datasets is discussed by Petar Afric [16]. FastText is one of the four classification techniques that are compared. The results show that FastText has a big impact even though the RoBERTa model generates the highest quality datasets. While SDP models trained on FastText-classified datasets do not outperform those trained on RoBERTa, they still yield insightful results. The study shows that while FastText can increase issue classification accuracy, its ability to improve SDP model performance is not as strong as that of the RoBERTa model.

The ensemble-based ML approaches [17] for software defect prediction from 2018 to 2021 are reviewed and evaluated in this review paper. Poor prediction still occurs despite advances because of problems with redundancy, correlation, and unbalanced data. Gaps in existing methods are exposed through the analysis of multiple viewpoints, evaluation criteria, and ML techniques. To overcome these obstacles and boost prediction performance in software defect detection, the paper promotes strong hyperparameter optimization, improved feature engineering, and the creation of stacking and averaging models.

Using ML [18] tackles the problem of class imbalance in software defect prediction (SDP). Several important conclusions are drawn from the research's systematic evaluation of 27 datasets, 7 classifiers, 7 input metrics, and 17 imbalanced learning techniques. Low imbalance is present in most datasets, which has little impact on traditional learning. On the other hand, performance is severely hampered by moderate to high imbalance. In this case, imbalanced learning may be advantageous, though outcomes may differ. The classifier type has the biggest effect on performance; input metrics have less of an impact. The imbalanced learning method comes in second. For moderately to highly imbalanced datasets, the study suggests using imbalanced learning. To prevent unfavorable outcomes, it is important to carefully choose the classifier-method combinations.

A greedy Extractive Summarization algorithm [19] enhanced by Variable Neighborhood Search (VNS) is used to summarize scientific articles from arXive and PubMed. Sentences with high TFIDF values are given priority by the algorithm, which also adjusts document frequency for TFIDF vectorization. It attains ROUGE-1/ROUGE-2 scores of 0.40/0.13 on PubMed and 0.43/0.12 on arXive, which are on par with the performance of cutting-edge models that make use of sophisticated neural networks and substantial computing power. This method, in contrast to these sophisticated models, is based on simple statistical inference, showing that less complex methods can still yield high-quality summaries.

## CHAPTER 4

### METHODOLOGY

The methodology encompasses several key stages aimed at effectively analyzing Java code for bug prediction. The stages are described in a detailed manner in the coming subsections as follows:

#### 4.1 Corpus Generation from using AST

The Python library `javalang` is used to represent the Java code in a tree-like structure that is the AST of the code. The Java code is taken from different Java projects described in Section 4.1. The `Javalang` library may be obtained from <https://github.com/c2nes/javalang>. There are two components in it: a lexer and a parser made specifically for Java [13]. Within the AST, each node corresponds to a specific construct such as `MethodDeclaration`, `IfStatement`, or `VariableAccess`, pinpointing occurrences within the source code. As a result, the AST facilitates the generation of a comprehensive corpus for each Java project. This corpus is used for fine-tuning the pre-trained TF-IDF, Word2Vec, Doc2Vec and FastText models.

#### 4.2 Generation of Sequence Tokens

The categories of AST nodes selected as tokens are control flow nodes, class declarations, and method invocations, which are also depicted in Table. 1. A new sequence token file is created for every version of the Java project (for example, Ant 1.5), and when any of the selected tokens in the table is detected within the corpus generated by the AST, that token is appended to the sequence token file. This procedure iterates for every version of the Java project, thereby composing the sequence tokens. These tokens are subsequently utilized as input for the models to generate embeddings.

#### 4.3 Fine Tuning of Pre-Trained Model

Transfer learning is employed by importing Word2Vec and TF-IDF models from Gensim and Scikit-Learn libraries, respectively. The models are trained on the corpus generated by AST for each Java project. These trained models are fed with tokens to generate the embeddings.

TF-IDF Vectorizer, imported from scikit-learn, is trained on the corpus. The vectorizer is fitted to the data using ‘fit-transform()’, analyzing text, constructing vocabulary, and calculating TF-IDF scores. The resulting sparse matrix represents documents, words, and TF-IDF scores, forming the trained TF-IDF model.

The Word2Vec model is imported from the Gensim library to train on a corpus generated by AST. Specific parameters are used to initialize and train the model, such as a vector size of 100 words, a window size of 5, a minimum count of 5 and an epoch of training the model that is equal to 10.

The Doc2Vec model is imported and trained based on a corpus produced by AST. Document embeddings which represent each document in a vector space are learned by it. The key parameters to initialize this model are: vector size (100), window size (5) and minimum word count (5). With 10 epochs, the model iterates over the dataset 10 times for training. Utilizing 4 CPU cores speeds up the process.

For FastText, we have initialized the imported model with parameters including vector size (100), window size (5), and minimum word count (5). The model undergoes 10 training epochs, utilizing 4 CPU cores for computational efficiency and accelerating the training process.

#### 4.4 Generation of Embeddings and workflow

After subjecting the pre-trained models to fine-tuning with the corpus generated by AST, the sequence tokens extracted from each version of the project are then inputted into the trained models. By doing so, the models are able to produce embeddings that reflect the underlying contextual information embedded within the code.

The whole process, as depicted in Fig. 4.1, goes by training a DL model using the vector representations obtained from the pre-trained models. This model is geared towards executing a specific task of defect prediction. The training process starts by inputting the embeddings into the model and iteratively refining model parameters to enhance performance.

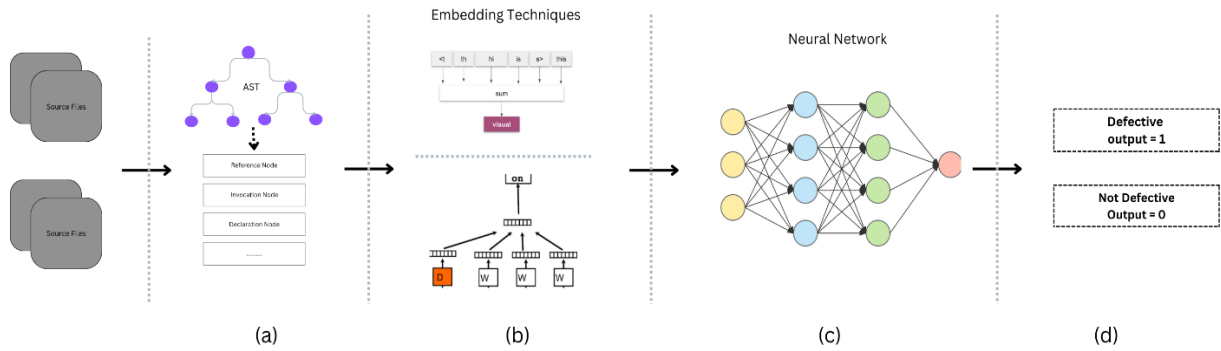


Figure 4.1. The process of defect prediction (a) Parsing the java code using AST. (b) Creating vectors using embedding techniques. (c) Training of DL model (d) Performing defect prediction

#### 4.5 Comparison of Techniques

The output of the trained DL model assigns "1" for bugs detected and "0" for bug-free software. After getting the final output, a comparative analysis is conducted to evaluate the embeddings based on the evaluation metrics.

This step assesses which embedding technique contributes more effectively to the model's performance. The important performance indicators used in this thesis are described further in section 5.2. Each of the embedding techniques is compared to one another on common terms based on evaluation metrics.



## CHAPTER 5

### Experimental Setup

This section outlines the dataset utilized, the baseline DL model selected for comprehensive analysis, the hyperparameter setting, as well as the evaluation measures included in the experiment.

#### 5.1 Dataset Used

The dataset used in this research is a set of 10 open-source Java projects that are taken from the PROMISE repository. The list of Java projects is given with their descriptions in Table 5.1.

Table 5.1 Description of project along with their version

Projects	Versions	Description
Ant	1.5, 1.6, 1.7	Java tool for managing processes, targets, and dependencies.
Camel	1.2, 1.4, 1.6	Open-source Java framework simplifying integration with EIPs, diverse transports, APIs.
Ivy	1.4, 2.0	Ivy, a sub-project of Apache Ant, resolves project dependencies using external XML files and downloads resources from repositories.
jEdit	4.0, 4.1, 4.2, 4.3	jEdit offers native syntax highlighting for more than 200 file formats, extendable via XML. Supports UTF-8 and various encodings with robust folding and wrapping features.
Log4j	1.0, 1.1, 1.2	log4j is integral to the Apache Logging Services Project, offering dependable, open-source logging utilities for diverse application needs.
lucene	2.0, 2.2, 2.4	Lucene is a Java-based, high-performance search engine library ideal for applications needing structured or full-text search, faceting, and more.
poi	2.0, 2.5, 3.0	POI, an open-source Java library, facilitates creation and manipulation of Microsoft Office file formats, enabling operations such as creation, modification, and reading.
synapse	1.1, 1.2	Synapse is analytics service merging data warehousing and Big Data analytics, offering flexible querying options with scalable resources.
xalan	2.4, 2.5, 2.6, 2.7	Xalan-Java functions as an XSLT processor, converting XML documents into various formats such as HTML, text, or other XML document types.
xerces	1.2, 1.3	It incorporates the Xerces Native Interface (XNI), offering a highly modular and programmable framework for building parser components and configurations.

#### 5.2 Evaluation Measure

A variety of evaluation metrics were employed as assessment measures to provide a full examination of the model's performance across varied criteria. In the following equations, there are several key terms, which are mentioned below:

1. True Positive (TP): Instances correctly classified as positive by the model.
2. False Positive (FP): Instances incorrectly classified as positive by the model.
3. True Negative(TN): Instances correctly classified as negative by the model.
4. False Negative(FN): Instances incorrectly classified as negative by the model.

Precision is the ratio of correctly predicted positive outcomes to all the predicted positive outcomes by the model.

$$Precision = \frac{TP}{TP+FP}$$

Recall measures the proportion of actual positive cases that were correctly identified by the model.

$$Recall = \frac{TP}{TP+FN}$$

The F1 Score is the harmonic mean of precision and recall. It is given as following.

$$F1\ Score = \frac{2*Precision*Recall}{Precision+Recall}$$

Accuracy is the ratio of correctly classified instances (both positives and negatives) to the total number of instances.

$$Accuracy = \frac{TP+TN}{TN+FP+TP+FN}$$

MCC is a measure of the quality of binary classifications. It is especially useful for evaluating models on imbalanced datasets.

$$MCC = \frac{TP * TN - FN * FP}{\sqrt{(TN + FN)(FP + TP)(TN + FP)(FN + TP)}}$$

FNR is the proportion of actual positive instances that were incorrectly classified as negative.

$$FNR = \frac{FN}{TP + FN}$$

FPR is the proportion of actual negative instances that were incorrectly classified as positive.

$$FPR = \frac{FP}{TN + FP}$$

TNR, also known as Specificity, is the proportion of actual negative instances that were correctly identified by the model.

$$TNR = \frac{TN}{TN + FP}$$

These metrics are fundamental in evaluating the performance of classification models, particularly in distinguishing between the different types of errors and successes the model makes.

### 5.3 Hyperparameter Settings

Training spanned 200 epochs to ensure comprehensive data learning. For ANN architectures, a sigmoid activation function was utilized throughout the layers, while rectified linear unit (ReLU) activation was applied in dense layers of RNNs (LSTM and GRU) with sigmoid activation in the output layer. Binary crossentropy served as the loss function across all models, optimized by the Adam optimizer. A batch size of 32 was chosen for computational efficiency and model stability. Two dense layers with 64 and 32 neurons, respectively, were employed to capture intricate data patterns. For CNN architecture, 1D convolutional layers were leveraged to capture spatial dependencies in sequential software data, thereby enhancing overall model performance.

## CHAPTER 6

### RESULT

In this section, the performance of all the embedding techniques such as Word2Vec, Fast Text, TF-IDF and Doc2Vec across the four discussed DL models is presented. Tables [6.1]–[6.16] contain the projects on which the models are trained, along with the mean values of accuracy (Acc.), precision (Prec.), F1-score (F1), and MCC is calculated from various versions of the same projects. For instance, the Lucene project had versions 2.0, 2.2, and 2.4. Training on version 2.0 and testing on version 2.2 yielded an accuracy of 0.63, while training on version 2.2 and testing on version 2.4 resulted in an accuracy of 0.61. The mean accuracy, calculated as 0.62, is included in the table. Detailed metrics are available at <https://github.com/GauravSharma171691/Results-Word-TFIDF>. Fig. 6.1 depicts mean FPR and FNR values for different embeddings TF-IDF (T) and Word2Vec (W), Doc2Vec(D) and FastText(F) across different project versions.

With the help of tables, we can see that models with Doc2Vec outperforms other embeddings exhibiting superior performance compared to models those leveraging other embeddings. The order of comparison is Doc2Vec, FastText, TF-IDF and Word2Vec (Doc2Vec being the best among others). Models with Doc2Vec embeddings have higher precision, F1-score, accuracy, MCC, and TNR. Higher values of Precision, F1 Score, Accuracy, and MCC show that the model is better and effective at classification tasks. Table (6.17) and (6.18) illustrates the average performance of DL models when employed with embeddings, respectively. The tables clearly describes the average performance of embedding techniques based on key performance metrics such as Accuracy, Precision, MCC, Recall and TNR.

The performance of the models trained in this study heavily depends on various aspects of the data it is trained on.

1. Too small dataset can lead to poor performance of model on unseen data.
2. Too large dataset may create models with many parameters affecting the deployment of model.

In this research, within-project defect prediction (WPDP) is used. In WPDP, the models are trained on a version of the project and tested on another subsequent version of the same project. This situation's performance can be improved by using cross-project defect prediction (CPDP) [15], where the model can be trained on one project and tested on another project.

Table 6.1 CNN based model using Word2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2372	0.9725	0.3805	0.2406	0.0619
Camel	0.1877	0.9927	0.3148	0.1954	0.0694
Ivy	0.2625	0.9674	0.413	0.2751	0.0142
Jedit	0.1311	0.9567	0.2203	0.1397	0.1053
log4j	0.6498	0.9705	0.7357	0.6623	0.0819
lucene	0.621	0.9907	0.7633	0.6235	0.1071
poi	0.4691	0.9877	0.5944	0.4752	0.0263
synapse	0.2741	0.9819	0.4286	0.28	0.0161
xalan	0.6484	0.9977	0.7608	0.6475	0.0266
xerces	0.1667	0.9792	0.2848	0.169	0.0744

Table 6.2 CNN based model using TF-IDF Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2517	0.9879	0.4012	0.2802	0.0861
Camel	0.2854	0.4644	0.3357	0.7178	0.1929
Ivy	0.0835	0.975	0.1538	0.1006	0.0085
Jedit	0.1993	0.9293	0.3125	0.4179	0.2486
log4j	0.6579	0.9825	0.7496	0.676	0.0746
lucene	0.6144	0.9906	0.7598	0.6159	0.068
poi	0.4785	0.9873	0.5972	0.4819	0.0504
synapse	0.2576	0.9833	0.4083	0.2565	0.1112
xalan	0.6489	0.9975	0.761	0.6491	0.0065
xerces	0.1535	0.9846	0.2656	0.1955	0.069

Table 6.3 CNN based model using FastText Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.4987	0.3736	0.3977	0.7529	0.4179
Camel	0.2425	0.2745	0.2563	0.6414	0.1317
Ivy	0.224	0.1165	0.206	0.683	0.159
Jedit	0.3228	0.5698	0.3824	0.8101	0.4624
log4j	0.7625	0.4234	0.5341	0.5549	0.2515
lucene	0.6178	0.7533	0.6785	0.6094	0.2866
poi	0.4323	0.4787	0.3694	0.4584	0.0512
synapse	0.4643	0.3535	0.4088	0.6581	0.3134
xalan	0.6182	0.3705	0.4786	0.4939	0.0302
xerces	0.2167	0.2001	0.208	0.7751	0.1158

Table 6.4 CNN based model using Doc2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.4686	0.3739	0.3976	0.7486	0.3599
Camel	0.2266	0.2845	0.2262	0.638	0.0782
Ivy	0.2501	0.1284	0.1834	0.9087	0.1802
Jedit	0.3199	0.5369	0.3809	0.8114	0.4525
log4j	0.8582	0.4013	0.59	0.5453	0.3196
lucene	0.6948	0.5488	0.6134	0.5508	0.2084
poi	0.4822	0.4647	0.3935	0.4459	0.0711
synapse	0.4746	0.3679	0.413	0.6691	0.2641
xalan	0.6705	0.3718	0.497	0.5306	0.1161
xerces	0.3145	0.2434	0.2743	0.8227	0.2625

Table 6.5 ANN based model using Word2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.23765	0.9785	0.3817	0.23975	0.0579
Camel	0.1859	0.9927	0.31255	0.18715	0.04575
Ivy	0.0856	0.9714	0.1574	0.1333	0.0347
Jedit	0.13125	0.956733	0.22105	0.145575	0.0818
log4j	0.6506	0.97055	0.73485	0.65965	0.1254
lucene	0.614	0.99395	0.7589	0.6131	0.021
poi	0.29847	0.98767	0.5934	0.3037	0.0409
synapse	0.2769	0.9818	0.432	0.29	0.0269
xalan	0.6489	0.99767	0.7612	0.64853	0.02
xerces	0.1661	0.9792	0.284	0.1655	0.1318

Table 6.6 ANN based model using TF-IDF Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2465	0.99155	0.3945	0.2595	0.0378
Camel	0.1893	0.9939	0.3179	0.2075	0.05895
Ivy	0.0833	0.975	0.1535	0.0985	0.0136
Jedit	0.1646	0.9509	0.2744	0.4367	0.1767
log4j	0.7612	0.87955	0.81275	0.83605	0.2271
lucene	0.61305	0.99405	0.75835	0.6116	0.0324
poi	0.47705	0.9866	0.59737	0.480033	0.07843
synapse	0.2794	0.9733	0.4324	0.3468	0.0867
xalan	0.4724	0.9974	0.64185	0.47295	0.0206
xerces	0.3145	0.2434	0.2743	0.8227	0.2625

Table 6.7 ANN based model using FastText Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.3913	0.710155	0.462935	0.71735	0.24405
Camel	0.263	0.60663	0.36692	0.62095	0.2219
Ivy	0.145	0.600375	0.20712	0.487125	0.07727
Jedit	0.34135	0.655233	0.433403	0.7447	0.46933
log4j	0.6819	0.6343	0.65695	0.60955	0.2042
lucene	0.66675	0.678015	0.66205	0.592525	0.21563
poi	0.3086	0.70963	0.393605	0.452278	0.09813
synapse	0.3661	0.63423	0.4385	0.62483	0.2282
xalan	0.4917	0.531435	0.49385	0.5211	0.09715
xerces	0.1938	0.416095	0.25898	0.64781	0.1149

Table 6.8 ANN based model using Doc2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.58175	0.426203	0.47462	0.824787	0.47091
Camel	0.23525	0.429635	0.294722	0.737234	0.204105
Ivy	0.18125	0.147125	0.153888	0.8735	0.119503
Jedit	0.34563	0.635923	0.42873	0.823402	0.52774
log4j	0.81975	0.471345	0.595833	0.60313	0.31713
lucene	0.71305	0.68333	0.69774	0.64546	0.34254
poi	0.477	0.552714	0.42744	0.508932	0.11932
synapse	0.6471	0.210795	0.318513	0.813402	0.410025
xalan	0.6718	0.58441	0.651973	0.52783	0.05844
xerces	0.2903	0.273925	0.29138	0.803965	0.19661

Table 6.9 GRU based model using Word2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2553	0.95045	0.39378	0.251533	0.06643
Camel	0.18565	0.9927	0.31158	0.1867	0.02966
Ivy	0.0818	0.939	0.14857	0.153125	0.03414
Jedit	0.197133	0.896767	0.295867	0.448533	0.12959
log4j	0.67715	0.7856	0.72048	0.7759	0.23524
lucene	0.61155	0.99345	0.75605	0.6091	0.0503
poi	0.4681	0.987667	0.629733	0.535	0.17515
synapse	0.2394	0.93085	0.3775	0.2768	0.08615
xalan	0.6498	0.997667	0.783167	0.646733	0.01925
xerces	0.2095	0.95415	0.31398	0.2175	0.06698

Table 6.10 GRU based model using TF-IDF Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2286	0.9801	0.3628	0.3188	0.11807
Camel	0.1719	0.99255	0.28655	0.173	0.0339
Ivy	0.0874	0.9778	0.1601	0.16955	0.05594
Jedit	0.15675	0.945	0.2589	0.3802	0.10366
log4j	0.65155	0.97995	0.78135	0.6207	0.132
lucene	0.61255	0.99405	0.75535	0.6101	0.04795
poi	0.507367	0.964967	0.6639	0.507867	0.1968
synapse	0.23835	0.9837	0.3816	0.2898	0.0271
xalan	0.6485	0.997233	0.782667	0.648067	0.0138
xerces	0.207233	0.96355	0.32343	0.217133	0.0532

Table 6.11 GRU based model using FastText Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2666	0.6534	0.3569	0.6556	0.0863
Camel	0.1971	0.7987	0.3597	0.2689	0.1536
Ivy	0.1623	0.8387	0.2815	0.4748	0.2967
Jedit	0.1641	0.5653	0.1914	0.7538	0.2083
log4j	0.7282	0.8651	0.7139	0.7629	0.197
lucene	0.7669	0.9507	0.8447	0.7669	0.1992
poi	0.5579	0.6492	0.5386	0.5145	0.244
synapse	0.3192	0.7649	0.3753	0.4043	0.1397
xalan	0.7192	0.8525	0.7533	0.6544	0.1554
xerces	0.1665	0.5433	0.1962	0.6151	0.2302

Table 6.12 GRU based model using Doc2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.3351	0.5547	0.4039	0.6328	0.1545
Camel	0.2845	0.9058	0.4487	0.4524	0.1679
Ivy	0.2376	0.7169	0.3129	0.7355	0.2408
Jedit	0.2021	0.7076	0.3096	0.6488	0.1922
log4j	0.7044	0.8322	0.7156	0.6494	0.1301
lucene	0.7656	0.9061	0.8285	0.7551	0.1655
poi	0.6186	0.5968	0.5683	0.5071	0.2089
synapse	0.3331	0.8734	0.4129	0.5108	0.1443
xalan	0.713	0.9174	0.762	0.722	0.175
xerces	0.469	0.563	0.4505	0.6631	0.2409

Table 6.13 LSTM based model using Word2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2285	0.9228	0.3594	0.3143	0.0362
Camel	0.1679	0.992	0.2821	0.2823	0.0501
Ivy	0.1073	0.9757	0.1602	0.2648	0.0046
Jedit	0.1295	0.866	0.2042	0.3333	0.1122
log4j	0.6645	0.9399	0.6862	0.6518	0.1093
lucene	0.6012	0.9879	0.7322	0.6159	0.0686
poi	0.4701	0.983	0.612	0.4824	0.0824
synapse	0.2368	0.962	0.3684	0.3585	0.0372
xalan	0.6489	0.9806	0.781	0.6464	0.0213
xerces	0.2166	0.9641	0.3469	0.3786	0.0946

Table 6.14 LSTM based model using TF-IDF Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2219	0.9065	0.3161	0.3846	0.1122
Camel	0.1807	0.8854	0.2839	0.2542	0.0645
Ivy	0.1683	0.9502	0.2516	0.1551	0.0936
Jedit	0.1831	0.8643	0.2462	0.4328	0.2433
log4j	0.5848	0.9414	0.6762	0.5963	0.0873
lucene	0.6259	0.9249	0.7296	0.6285	0.1041
poi	0.4924	0.9724	0.6368	0.5878	0.1325
synapse	0.2369	0.8756	0.3501	0.2561	0.1504
xalan	0.4842	0.8446	0.5041	0.7038	0.1024
xerces	0.1876	0.7241	0.2784	0.2987	0.1061

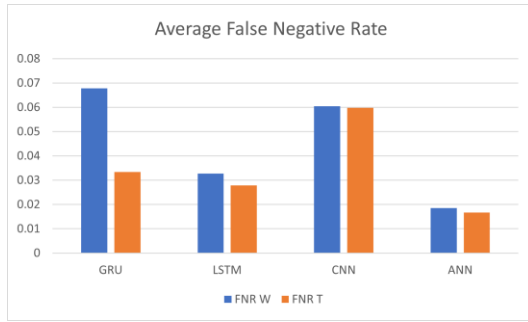
Table 6.15 LSTM based model using FastText Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2109	0.3512	0.2294	0.3765	0.3379
Camel	0.1659	0.6266	0.2301	0.2427	0.1857
Ivy	0.0998	0.3725	0.1448	0.1598	0.2879
Jedit	0.1329	0.3082	0.1602	0.3316	0.3232
log4j	0.6487	0.6905	0.6551	0.8545	0.2373
lucene	0.6102	0.7378	0.6386	0.9197	0.295
poi	0.4664	0.4828	0.4078	0.9751	0.2084
synapse	0.2109	0.2174	0.2082	0.3948	0.2456
xalan	0.4783	0.6247	0.4908	0.7037	0.236
xerces	0.4294	0.4007	0.3684	0.7348	0.2098

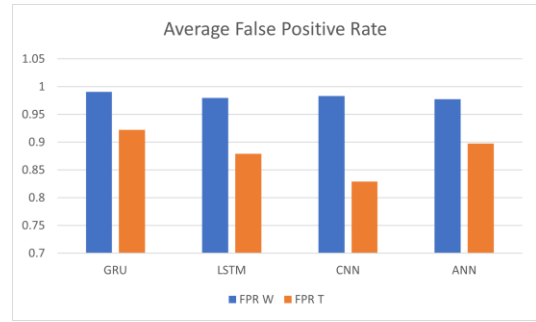
Table 6.16 LSTM based model using Doc2Vec Embedding

Project	Precision	Recall	F1 Score	Accuracy	MCC
Ant	0.2515	0.4126	0.3011	0.5642	0.4583
Camel	0.191	0.2532	0.2146	0.2791	0.2243
Ivy	0.1842	0.2301	0.2052	0.2623	0.2945
Jedit	0.1462	0.2745	0.1757	0.4303	0.3669
log4j	0.6795	0.6874	0.6825	0.8457	0.2205
lucene	0.6132	0.7358	0.6539	0.9197	0.3069
poi	0.2717	0.4336	0.2932	0.4154	0.2196
synapse	0.2344	0.2408	0.2267	0.3948	0.3867
xalan	0.4753	0.5638	0.4908	0.7072	0.2291
xerces	0.2222	0.2499	0.2083	0.3645	0.1538

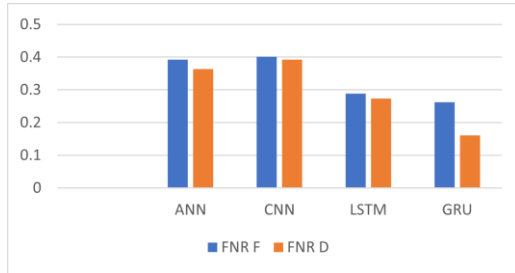




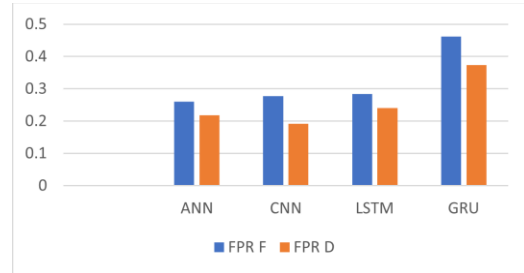
(a)



(b)



(c)



(d)

Figure 6.1 Average values of (a) False Negative Rate using Word2Vec and TF-IDF (b) False Positive Rate using Word2Vec and TF-IDF (c) False Negative Rate using FastText and Doc2Vec (d) False Positive Rate using FastText and Doc2Vec.

Table 6.17 Average performance of DL models with Word2Vec and TF-IDF embeddings

Embedding	Model	Precision	F1Score	Accuracy	MCC
Word2Vec	GRU	0.3809	0.4976	0.386	0.0668
	LSTM	0.3795	0.4927	0.3836	0.0506
	CNN	0.392	0.5121	0.3978	0.061
	ANN	0.3826	0.4989	0.3893	0.0551
TF-IDF	GRU	0.3947	0.5099	0.4259	0.0879
	LSTM	0.3998	0.5167	0.4618	0.11
	CNN	0.4047	0.5166	0.5087	0.0974
	ANN	0.4004	0.5153	0.4584	0.0848

Table 6.18 Average performance of DL models with FastText and Doc2Vec embeddings

Embedding	Model	Precision	Accuracy	MCC	TNR
FastText	ANN	0.493	0.645	0.254	0.771
	CNN	0.473	0.626	0.219	0.71
	GRU	0.453	0.603	0.196	0.606
	LSTM	0.382	0.54	0.282	0.627
Doc2Vec	ANN	0.515	0.695	0.281	0.789
	CNN	0.493	0.646	0.241	0.813
	GRU	0.479	0.668	0.212	0.671
	LSTM	0.397	0.597	0.286	0.644

## CHAPTER 7

### CONCLUSION AND FUTURE WORK

The performance of four different word embedding methods TF-IDF, Word2Vec, FastText, and Doc2Vec was assessed in the context of SDP using various DL methods. Among these, Doc2Vec demonstrated the best performance, followed by FastText, TF-IDF, and Word2Vec. The evaluation aimed to determine the effectiveness of these embedding techniques in enhancing classification accuracy and reliability across a wide range of criteria.

There are diverse prospects that can be exploited as a result of this research. Firstly, other advanced and specific embeddings like BERT, Code-BERT, RoBERTa, ELMO, XLNet can be considered. These techniques have manifested improved abilities to capture semantic relationships with higher accuracy that could further improve the performance of SDP models.

In addition, in future studies on software engineering, the PROMISE dataset may be used but NASA's dataset should be included in its investigation. The latter would increase the size of a corpus which would possibly engender better and more universal models. One more direction is to use CPDP rather than WPDP. Due to using multiple projects data for training DL models in contrast to WPDP methodology, CPDP approach enhances the ability of deep learning (DL) models to generalize across different contexts.

This is an avenue that can be followed by hybrid models incorporating the strengths of different embedding techniques and DL architectures so as to capture more diverse features and relationships within the data. Also, it might be beneficial to test these improved versions across various languages or domains with expected results being more robust SDPs with enhanced adaptability and resilience. Continuous benchmarking against new models and embedding techniques is crucial in the rapidly evolving field of NLP to maintain effectiveness and relevance. While this study identified Doc2Vec as the most effective embedding method tested, significant potential for improvement remains through exploring advanced embedding techniques, using larger and more diverse datasets, and developing hybrid models.

## REFERENCES

- [1] Sharma, T., Jatain, A., Bhaskar, S., Pabreja, K. (2023). Ensemble ML Paradigms in Software Defect Prediction. *Procedia Computer Science*, 218, 199-209. <https://doi.org/10.1016/j.procs.2023.01.002>
- [2] Malhotra, R., Singh, P. (2023). Recent advances in DL models: a systematic literature review. *Multimed Tools Appl*, 82, 44977–45060. <https://doi.org/10.1007/s11042-023-15295-z>
- [3] Mambina, I.S., Ndibwile, J.D., Michael, K.F.: 'Classifying Swahili Smishing Attacks for Mobile Money Users: A Machine-Learning Approach'. In: *IEEE Access*, 83061--83074 (2022) DOI: 10.1109/ACCESS.2022.3196464
- [4] Es-Sabery, F., Es-Sabery, I., Hair, A., Sainz-De-Abajo, B., Garcia-Zapirain, B.: 'Emotion Processing by Applying a Fuzzy-Based Vader Lexicon and a Parallel Deep Belief Network Over Massive Data'. In: *IEEE Access* 10, 87870--87899 (2022)
- [5] Xie, Z., Liu, L., Wu, Y., Li, L., Zhong, L.: 'Learning TF-IDF Enhanced Joint Embedding for Recipe-Image Cross-Modal Retrieval Service'. In: *IEEE*, pp. 3304--3316 Publisher: IEEE. (2021)
- [6] Canales, L., Strapparava, C., Boldrini, E., Martínez-Barco, P. (2020). "Intensional Learning to Efficiently Build Up Automatically Annotated Emotion Corpora." *IEEE Transactions on Affective Computing*, 11(2), 335-347. DOI: 10.1109/TAFFC.2017.2764470.
- [7] Miholca, D.-L., Tomescu, V.-I., Czibula, G.: 'An in-depth Analysis of the Software Features' Impact on the Performance of DL-Based Software Defect Predictors'. *IEEE Access* 10, 64801--64818 (2022)
- [8] Nevendra, M., & Singh, P. (2022). A Survey of Software Defect Prediction Based on DL. *Archives of Computational Methods in Engineering*, 29, 5723-5748. <https://doi.org/10.1007/s11831-022-09787-8>
- [9] Wang, X., Lu, L., Wang, B., Shang, Y., Yang, H.: 'SDP via GIN with Hybrid Graphical Features'. *IEEE 22nd International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*(2023)
- [10] Tang, F., He, P.: 'SDP using Multi-scale Structural Information'. In: *ICCAI '23: Proceedings of the 2023 9th International Conference on Computing and Artificial Intelligence*, March 2023, pp. 548-556.
- [11] Liu, D., Jiang, H., Li, X., Ren, Z., Qiao, L., Ding, Z.: 'DPWord2Vec: Better Representation of Design Patterns in Semantics'. *IEEE Transactions on Software Engineering*, 48(4), 1228--1248 (2022)
- [12] Cao, B., Zhang, L., Peng, M., Qing, Y., Kang, G., Liu, J.: 'Web Service Recommendation via Combining Bilinear Graph Representation and xDeepFM Quality Prediction'. *IEEE Volume 20 Issue 2* 1078--1092 (2023)
- [13] Fan, G., Diao, X., Yu, H., Yang, K., & Chen, L. (2019). Software Defect Prediction via Attention-Based Recurrent Neural Network. *Scientific Programming*, Volume 2019, <https://doi.org/10.1155/2019/6230953>

- [14] Liang, H., Yu, Y., Jiang, L., \& Xie, Z. (2019). SEML: A Semantic LSTM Model for Software Defect Prediction. IEEE Access, 7, 83812-83824. <https://doi.org/10.1109/ACCESS.2019.2925313>
- [15] Bala, Y.Z., Samat, P.A., Sharif, K.Y., Manshor, N.: 'Improving Cross-Project SDP Method Through Transformation and Feature Selection Approach'. IEEE Access 11, 2318--2326 (2022). IEEE.
- [16] P. Afric, D. Vukadin, M. Silic, and G. Delac, "Empirical Study: How Issue Classification Influences Software Defect Prediction," IEEE Access, vol. 11, pp. 11732-11748, Feb. 2023.
- [17] T. Sharma, A. Jatain, S. Bhaskar, and K. Pabreja, "Ensemble Machine Learning Paradigms in Software Defect Prediction," Procedia Computer Science, vol. 218, pp. 199-209, Jan. 2023.
- [18] Q. Song, Y. Guo, and M. Shepperd, "A Comprehensive Investigation of the Role of Imbalanced Learning for Software Defect Prediction," IEEE Transactions on Software Engineering, vol. 45, no. 12, pp. May 15, 2018.
- [19] I. Akhmetov, A. Gelbukh, and R. Mussabayev, "Greedy Optimization Method for Extractive Summarization of Scientific Articles," IEEE Access, vol. 9, pp. 168141-168153, Dec. 2021.



**DELHI TECHNOLOGICAL UNIVERSITY**  
(Formerly Delhi College of Engineering)  
Shahbad Daultapur, Main Bawana Road, Delhi-42

**PLAGIARISM VERIFICATION**

Title of the Thesis Comparative Analysis of Word Embedding  
Techniques on Software Defect Prediction

Total Pages 29 Name of the Scholar Gaurav Sharma

Supervisor (s)

(1) Priya Singh

(2) \_\_\_\_\_

(3) \_\_\_\_\_

Department Software Engineering

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: Turnitin Similarity Index: 11%, Total Word Count: 7558

Date: 22<sup>nd</sup> May 2024

Gaurav Sharma  
Candidate's Signature

Priya  
Signature of Supervisor(s)

PAPER NAME

**Gaurav\_Plug.docx**

WORD COUNT

**7558 Words**

CHARACTER COUNT

**42631 Characters**

PAGE COUNT

**29 Pages**

FILE SIZE

**794.1KB**

SUBMISSION DATE

**May 21, 2024 11:51 PM GMT+5:30**

REPORT DATE

**May 21, 2024 11:52 PM GMT+5:30**

### ● 11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 6% Internet database
- 5% Publications database
- Crossref database
- Crossref Posted Content database
- 9% Submitted Works database

### ● Excluded from Similarity Report

- Bibliographic material
- Quoted material

## ● 11% Overall Similarity

Top sources found in the following databases.

- 6% Internet database
- Crossref database
- 9% Submitted Works database
- 5% Publications database
- Crossref Posted Content database

## TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	<b>sigarra.up.pt</b> Internet	2%
2	<b>National Institute of Technology, Rourkela on 2024-01-16</b> Submitted works	<1%
3	<b>Boyang Liu, Guozheng Rao, Xin Wang, Li Zhang, Qing Cong. "DE3TC: De..."</b> Crossref	<1%
4	<b>origin.geeksforgeeks.org</b> Internet	<1%
5	<b>cris.maastrichtuniversity.nl</b> Internet	<1%
6	<b>Cranfield University on 2023-08-17</b> Submitted works	<1%
7	<b>cdn.techscience.cn</b> Internet	<1%
8	<b>National College of Ireland on 2023-08-25</b> Submitted works	<1%

9	Venkata Vara Prasad D, Lokeswari Y Venkataramana, P. Senthil Kumar,...	<1%
	Crossref	
10	University of Strathclyde on 2023-08-31	<1%
	Submitted works	
11	export.arxiv.org	<1%
	Internet	
12	University of Houston, Downtown on 2024-03-28	<1%
	Submitted works	
13	University of Wollongong on 2023-12-05	<1%
	Submitted works	
14	Tilburg University on 2024-05-20	<1%
	Submitted works	
15	fastercapital.com	<1%
	Internet	
16	University of Houston, Downtown on 2024-03-31	<1%
	Submitted works	
17	Anna University on 2020-04-13	<1%
	Submitted works	
18	students.takelab.fer.hr	<1%
	Internet	
19	link.springer.com	<1%
	Internet	
20	ndl.ethernet.edu.et	<1%
	Internet	



21	<b>bsj.uobaghdad.edu.iq</b> Internet	<1%
22	<b>Ngee Ann Polytechnic on 2023-08-15</b> Submitted works	<1%
23	<b>Purdue University on 2023-12-27</b> Submitted works	<1%
24	<b>Liverpool John Moores University on 2024-05-13</b> Submitted works	<1%
25	<b>University of Leeds on 2018-05-02</b> Submitted works	<1%
26	<b>qdosd.squiz.cloud</b> Internet	<1%
27	<b>theses.gla.ac.uk</b> Internet	<1%
28	<b>hindawi.com</b> Internet	<1%
29	<b>Bournemouth University on 2024-05-21</b> Submitted works	<1%
30	<b>Mudasir Ahmad Wani, Mohammad ELAffendi, Patrick Bours, Ali Shariq ...</b> Crossref	<1%
31	<b>Sabancı Universitesi on 2006-12-26</b> Submitted works	<1%
32	<b>Southern New Hampshire University - Continuing Education on 2024-0...</b> Submitted works	<1%

33	Xuanye Wang, Lu Lu, Boye Wang, Yudong Shang, Hao Yang. "Software ... Crossref	<1%
34	Colorado State University, Global Campus on 2022-12-04 Submitted works	<1%
35	National College of Ireland on 2022-08-14 Submitted works	<1%
36	Rana Husni AlMahmoud, Bassam H. Hammo. "SEWAR: A corpus-based... Crossref	<1%
37	Subba Reddy Borra, Dasari Ramesh Gari Amrutha Nayana, Sripathi Srin... Crossref	<1%
38	The University of Manchester on 2010-09-10 Submitted works	<1%
39	journals.plos.org Internet	<1%
40	ntnuopen.ntnu.no Internet	<1%
41	research-collection.ethz.ch Internet	<1%
42	"Intelligent Natural Language Processing: Trends and Applications", Sp... Crossref	<1%
43	Oxford Brookes University on 2019-03-31 Submitted works	<1%
44	The Robert Gordon University on 2020-06-08 Submitted works	<1%

45	<b>Tilburg University on 2024-05-20</b> Submitted works	<1%
46	<b>University of Hertfordshire on 2023-12-04</b> Submitted works	<1%
47	<b>University of Malta on 2018-09-29</b> Submitted works	<1%
48	<b>University of Surrey on 2023-09-04</b> Submitted works	<1%
49	<b>docplayer.net</b> Internet	<1%
50	<b>Buqing Cao, Lulu Zhang, Mi Peng, Yueying Qing, Guosheng Kang, Jianx...</b> Crossref	<1%
51	<b>Universiti Sains Malaysia on 2015-06-25</b> Submitted works	<1%
52	<b>AlHussein Technical University on 2024-01-29</b> Submitted works	<1%
53	<b>Asia Pacific University College of Technology and Innovation (UCTI) on...</b> Submitted works	<1%

## DECLARATION

We/I hereby certify that the work which is presented in the Major Project-II/Research Work entitled Comparative Analysis of Word Embedding Techniques in Software Defect Prediction in fulfillment of the requirement for the award of the Degree of Bachelor/Master of Technology in Data Science and submitted to the Department of Software Engineering, Delhi Technological University, Delhi is an authentic record of my/our own, carried out during a period from 2022, under the supervision of Priya Singh.

The matter presented in this report/thesis has not been submitted by us/me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/communicated in SCI/ SCI expanded/SSCI/Scopus indexed journal OR peer reviewed Scopus indexed conference with the following details:

Title of the Paper: Comparative Study: Word2Vec vs. TF-IDF in Software Defect Prediction

Author names (in sequence as per research paper): Gaurav Sharma, Priya Singh

Name of Conference/Journal: 2<sup>nd</sup> International Conference on Data Science and Network Engineering

Conference Dates with venue (if applicable): 12 - 13 July 2024

Have you registered for the conference (Yes/No)? Yes

Status of paper (Accepted/Published/Communicated): Accepted

Date of paper communication:

Date of paper acceptance: 14<sup>th</sup> May 2024

Date of paper publication:

Gaurav Sharma

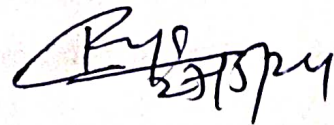
2222/DSCL/05

Gaurav Sharma

Student(s) Roll No., Name and Signature

## SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I, further certify that the publication and indexing information given by the students is correct.



Place: Delhi

Supervisor Name and Signature

Date: 22<sup>nd</sup> May 2024

Miss. Priya Singh

NOTE: PLEASE ENCLOSE RESEARCH PAPER ACCEPTANCE/  
PUBLICATION/COMMUNICATION PROOF ALONG WITH SCOPUS INDEXING PROOF  
(Conference Website OR Science Direct in case of Journal Publication).



Gaurav Sharma <sharmagaurav171691@gmail.com>

---

## 2nd International Conference on Data Science and Network Engineering : Submission (146) has been edited.

3 messages

---

**Microsoft CMT** <email@msr-cmt.org>  
Reply-To: Microsoft CMT - Do Not Reply <noreply@msr-cmt.org>  
To: sharmagaurav171691@gmail.com

1 April 2024 at 19:27

Hello,

The following submission has been edited.

Track Name: Track 1: AI, ML, and DL

Paper ID: 146

Paper Title: Comparative Study: Word2Vec vs. TF-IDF in Software Defect Predictions”

**Abstract:**

Embeddings, renowned for their ability to capture semantic nuances, reduce dimensionality, and learn patterns from data, have become indispensable in various domains of machine learning and artificial intelligence. In the realm of Software Defect Prediction (SDP), the selection of an appropriate embedding technique holds paramount importance. This study scrutinizes the comparative efficacy of two prominent embedding methods, Word2Vec and TF-IDF, in the context of SDP tasks. The entire analysis was conducted on different sets of Java projects sourced from the open-source Promise repository. Through the training and evaluation of diverse deep learning models tailored for defect prediction in software, a comprehensive assessment was undertaken. Evaluation metrics, including Matthews Correlation Coefficient (MCC), Specificity, Accuracy, alongside additional performance indicators, were employed to assess the effectiveness of each embedding technique. The findings unequivocally reveal that TF-IDF consistently outperforms Word2Vec across multiple metrics, emphasizing its superior suitability for SDP tasks.

Keywords: Software Defect Prediction · Word2Vec · TF-IDF

Created on: Mon, 01 Apr 2024 13:55:53 GMT

Last Modified: Mon, 01 Apr 2024 13:57:00 GMT

**Authors:**

- [sharmagaurav171691@gmail.com](mailto:sharmagaurav171691@gmail.com) (Primary)
- [priya.singh.academia@gmail.com](mailto:priya.singh.academia@gmail.com)

Secondary Subject Areas: Not Entered

Submission Files: [Article\\_Title.pdf](#) (338 Kb, Mon, 01 Apr 2024 13:55:07 GMT)

Submission Questions Response: Not Entered

Thanks,  
CMT team.

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).

Microsoft Corporation  
One [Microsoft Way](#)  
[Redmond, WA 98052](#)

---





Gaurav Sharma <sharmagaurav171691@gmail.com>

## Decision (Accept) on Manuscript (Paper ID 146) of ICDSNE 2024

2 messages

Microsoft CMT <email@msr-cmt.org>

13 May 2024 at 22:20

Reply-To: Suyel Namasudra <suyel.namasudra@nita.ac.in>

To: Gaurav Sharma SHARMA <sharmagaurav171691@gmail.com>

Dear Gaurav Sharma SHARMA,

Thank you for submitting your manuscript (Paper ID 146) entitled "Comparative Study: Word2Vec vs. TF-IDF in Software Defect Predictions" for a possible publication in the Proceedings of the Second International Conference on Data Science and Network Engineering (ICDSNE 2024).

We have received the reports from the reviewers on your manuscript. Based on the received reviews, we are pleased to inform you that your manuscript has been accepted for further processing.

While preparing the Camera-Ready paper, you are requested to consider the following comments:

1. Please use a native English-speaking editor. Papers with less than excellent English will not be published even if technically perfect/accepted.
2. If there are any reviewers' comments, describe in a separate file how the comments of each reviewer are addressed.
3. The paper's title should be concise and as short as possible. Do not use any acronyms in the paper's title.
4. Make sure the Abstract succinctly describes the paper as it is used in abstracting and citation services. Keep the Abstract between 150 to 200 words.
5. Do not use any references in the Abstract.
6. Spell out each acronym the first time used in the body of the paper. Spell out acronyms in the Abstract only, if used there.
7. Include a list of four to six keywords after the Abstract, which are not used in the title.
8. Include a paragraph at the end of the Introduction describing the organization of the paper. Also, mention the point-wise contributions before this paragraph.
9. Make sure that the Conclusion succinctly summarizes the paper; it should not repeat phrases from the Introduction (such as 'This paper presents ...')! Keep the Conclusion to about 300 words. Do not use any references in the Conclusion.
10. Number the references sequentially as they are used in the text (not alphabetically). Each reference must be referred to in the text. Use recent references. Provide complete information for all references (authors' names, journal/conference name, pages, etc.; do not use et al. for authors).
11. Make sure all figures and tables are sequentially referred to in the body of the paper.
12. Clearly mention the email IDs of the corresponding authors.
13. No published materials (such as, figures, tables, etc.) should be used in the paper. All the contents should be original.
14. Please visit the following links to prepare the Camera-Ready paper:
  - \* SpringerNature's proceedings website ([bit.ly/3AVXpzc](https://bit.ly/3AVXpzc))
  - \* Word template for preparing the camera-ready paper ([bit.ly/3HGuQJH](https://bit.ly/3HGuQJH)) or LaTeX template for preparing the camera-ready paper ([bit.ly/3AWSYEd](https://bit.ly/3AWSYEd))

Please follow the below-mentioned steps to complete the Registration Process:

1. Deposit the Registration Fee to the following bank account as per the category. Kindly visit our Conference Website ([www.dsne.in](http://www.dsne.in)) to know the Registration Fee details. If you are doing an online transfer, please save a Screenshot of the online transaction and mention a comment like "Reg. Fee of Paper ID 146" during payment.

Beneficiary Name: ICDSNE  
Account Number: 41940623168  
IFSC Code: SBIN0011491  
SWIFT Code: SBININBB476  
Bank Name: State Bank of India  
Branch Name: NIT Agartala

2. Make a ZIP file that consists of the following files:

1. Camera-Ready Paper Source File
2. Camera-Ready Paper PDF
3. Supplementary file (if any)
4. High-resolution figures (if any) in .jpg, .png, or .tif format in a folder, namely Figures. The file name of the figures should be the figure number (like Figure 1).

3. Fill out the following form to complete the registration process:

---

< ICDSNE 2024

# SECOND INTERNATIONAL CONFERENCE ON DATA SCIENCE AND NETWORK ENGINEERING

📅 12-13 July, 2024

📍 Agartala, India.

alt="slide"/>

## INTERNATIONAL CONFERENCE ON Data Science and Network Engineering

Second International Conference on Data Science and Network Engineering (ICDSNE 2024) is being organized by the Department of Computer Science and Engineering, National Institute of Technology Agartala (NIT Agartala), India, in hybrid mode (online/in-person) on 12-13 July, 2024. This institute is an Institution of National Importance by Ministry of Education, Govt. of India, and it is located in Tripura, India. There are 4702 students enrolled in different undergraduate, post-graduate, and PhD programmes at NIT Agartala. This institute supports excellent teaching and research environments to produce leaders, who can make a difference in the world. ICDSNE 2024 is a non-profitable conference, and it provides an opportunity for researchers, academicians, and industry professionals to present their research work on data science and network engineering. The main objective of this conference is to bring together researchers and practitioners in the world working on data science and network engineering, so that they can share ideas, innovations, and recent trends in their respective areas to address real-time problems. ICDSNE 2024 will feature a range of presentations on the latest research activities, as well as stimulating talks, and keynote addresses. There are multiple tracks in the conference covering almost all the areas of data science and network engineering. The organizing committee is confident that this conference will provide a platform for researchers to share their ideas and to have future collaboration with different people across the world. All accepted, registered, and presented papers will be included in the Springer Book Series entitled "Lecture Notes in Networks and Systems", which is indexed in DBLP, Scopus, and many more.



To 3168

₹7,000

Pay again

Split with friends

✓ Completed

19 May 2024, 4:17 pm



State Bank of India  
4320



- Payment started  
4:17 pm
- ₹7,000 was debited  
4:17 pm
- ₹7,000 sent to 3168  
4:17 pm
- ✓ Payment completed  
4:17 pm

UPI transaction ID

414008726445

To: NATIONAL INSTITUTE OF  
TECHNOLOGY AGARTALA

....3168

From: ASHISH KUMAR (State Bank of  
India)

keshriashish07@oksbi

Google transaction ID

CICAgPDs66PETA

Powered by





## DECLARATION

We/I hereby certify that the work which is presented in the Major Project-II/Research Work entitled Comparative Analysis of Word Embedding Techniques in Software Defect Prediction in fulfillment of the requirement for the award of the Degree of Bachelor/Master of Technology in Data Science and submitted to the Department of Software Engineering, Delhi Technological University, Delhi is an authentic record of my/our own, carried out during a period from 2022, under the supervision of Priya Singh.

The matter presented in this report/thesis has not been submitted by us/me for the award of any other degree of this or any other Institute/University. The work has been published/accepted/communicated in SCI/SCI expanded/SSCI/Scopus indexed journal OR peer reviewed Scopus indexed conference with the following details:

Title of the Paper: Comparative analysis of FastText and Doc2Vec for semantic Feature oriented Software Defect Prediction  
Author names (in sequence as per research paper): Priya Singh, Gaurav Sharma  
Name of Conference/Journal: International conference on Intelligent computing and Communication Techniques  
Conference Dates with venue (if applicable): 28-29 June 2024  
Have you registered for the conference (Yes/No)? Yes  
Status of paper (Accepted/Published/Communicated): Accepted  
Date of paper communication:  
Date of paper acceptance: 9th May 2024  
Date of paper publication:

Gaurav Sharma  
2022/DSE/05  
Gaurav Sharma

Student(s) Roll No., Name and Signature

## SUPERVISOR CERTIFICATE

To the best of my knowledge, the above work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere. I, further certify that the publication and indexing information given by the students is correct.

Place: Delhi

Date: 22nd May 2024



Supervisor Name and Signature

Miss. Priya Singh

NOTE: PLEASE ENCLOSE RESEARCH PAPER ACCEPTANCE/PUBLICATION/COMMUNICATION PROOF ALONG WITH SCOPUS INDEXING PROOF (Conference Website OR Science Direct in case of Journal Publication).



Gaurav Sharma <sharmagaurav171691@gmail.com>

---

## International Conference on Intelligent Computing and Communication Techniques : Submission (543) has been created.

1 message

---

Microsoft CMT <email@msr-cmt.org>

5 May 2024 at 16:51

Reply-To: Microsoft CMT - Do Not Reply <noreply@msr-cmt.org>

To: sharmagaurav171691@gmail.com

Hello,

The following submission has been created.

Track Name: Artificial Intelligence

Paper ID: 543

Paper Title: Analyzing Software Defect Prediction: A Comparative Study of FastText and Doc2Vec Techniques

**Abstract:**

mbeddings are valued for their capacity to grasp semantic relation, streamlining dimensionality and discerning data patterns. These are used across various domains of machine learning and artificial intelligence. In the area of Software Defect Prediction, choosing an appropriate embedding technique is very important. This study aims to compare the effectiveness of two prominent word embedding methods that are FastText and Doc2Vec when applied for detecting bugs in softwares. The entire analysis is conducted by using different sets of Java projects taken from an open-source Promise repository. Through rigorous training and evaluation of several deep learning models, generally designed for defect detection in software, a comprehensive evaluation was conducted. Evaluation metrics, such as Matthews Correlation Coefficient, Specificity, and Accuracy, along with other important performance indicators, were used to assess the effectiveness of both the technique. The results indicate that Doc2Vec performs significantly better than FastText in terms of multiple metrics, depicting its superiority in predicting defects in the software.

Created on: Sun, 05 May 2024 11:20:55 GMT

Last Modified: Sun, 05 May 2024 11:20:55 GMT

**Authors:**

- [priya.singh.academia@gmail.com](mailto:priya.singh.academia@gmail.com) (Primary)
- [sharmagaurav171691@gmail.com](mailto:sharmagaurav171691@gmail.com)

Secondary Subject Areas: Not Entered

Submission Files: F\_D2Vec.pdf (298 Kb, Sun, 05 May 2024 11:15:20 GMT)

Submission Questions Response: Not Entered

Thanks,  
CMT team.

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).



Gaurav Sharma <sharmagaurav171691@gmail.com>

## Notification of acceptance of paper id 543

1 message

Microsoft CMT <email@msr-cmt.org>

9 May 2024 at 11:51

Reply-To: ICICCT 2024 <icicctcon@gmail.com>

To: Gaurav Sharma SHARMA <sharmagaurav171691@gmail.com>

Dear Dr./ Prof. Gaurav Sharma SHARMA,

Congratulations...

Your paper / article paper id 543: Analyzing Software Defect Prediction: A Comparative Study of FastText and Doc2Vec Techniques has been accepted for publication in International Conference on Intelligent Computing and Communication Techniques at JNU New Delhi, India.

Kindly save your paper by given paper id only (eg. 346.docx, 346.pdf, 346\_copyright.pdf)

Registration Link:

<https://forms.gle/mSsHa8GMLtMkWuaq8>

Please ensure the following before registration and uploading camera ready paper.

1. Paper must be in Taylor and Frances Format.

Template and copyright with author instruction are given in below link: [https://icicct.in/author\\_inst.html](https://icicct.in/author_inst.html)

2. Minimum 12 references should be cited in the paper and all references must be cited in the body. Please follow the template.

3. The typographical and grammatical errors must be carefully looked at your end.

4. Complete the copyright form (available at template folder).

5. The regular fee (Available in registration section) will be charged up to 6 pages and after that additional Rs.1000 for Indian authors / 10 USD for foreign authors per additional page will be charged.

6. Reduce the Plagiarism below 10% excluding references and AI Plagiarism 0%.

Each Illustration must include a caption and an alternative text description to assist print impaired readers ('Alt Text').

(Alt Text is mandatory for each Illustrations)

7. Certificate will be issued by the name of registered author (Single author only).

8. Certificates may be issued to all other authors on the extra payment of 1000/- INR per author.

9. Last Date of registration and uploading copyright and camera-ready copy: 31/05/2024.

10. Make a single payment which includes registration fee + Extra certificates fee + Extra page fees.

11. Permissions: Kindly make sure the permissions for each copyrighted artwork file have been cleared ahead of the submission, with the details listed in the Permission Verification form (attached). All permission grants must be submitted along with your final manuscript.

Figures: Please make sure no figures are missing, and all figures are high resolution

Tables: Please ensure that there are no missing tables, and the tables in your manuscript are not pasted as figures

Citation: Kindly ensure there are no missing citations in your manuscript

Registration Link: <https://forms.gle/mSsHa8GMLtMkWuaq8>

Registration Fee to be deposited in below account

Bank Account Details :

Indian Account Details:

Account Holder Name: EVEDANT Foundation

Account Number: 0674002190422900

IFSC Code: PUNB0067400

SWIFT Code: PUNBINBBGNM

Branch: Punjab National Bank, Navyug Market, Ghaziabad



Academic Excellence and Research Award

Academic Excellen

Academic Excellence  
and Research Award

# ICICCT

Apply for Academic Excellence and  
Research Award Now !!!

## INTERNATIONAL CONFERENCE

on

### Intelligent Computing and Communication Techniques

28<sup>th</sup>-29<sup>th</sup> June 2024, JNU, New Delhi

Hybrid (Online & Offline) Mode

**Registration Date has been  
extended till 05-06-2024**

~~10<sup>th</sup> May 2024~~

~~15<sup>th</sup> May 2024 (Extended)~~





# Transaction Successful

03:15 pm on 30 May 2024

## Paid to



Evedant Foundation

₹7,000

XXXXXXXXXXXX2900

Punjab National Bank



Transfer Details



Transaction ID

T2405301515021898830127

Debited from



XXXXXX4700

₹7,000

UTR: 415189120638

Powered by

