Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 335–346
DOI: 10.15439/2023F6950
ISSN 2300-5963 ACSIS, Vol. 35

Comparative Analysis of Word Embedding and Machine Learning Techniques for Classification of Software Developer Communications on Gitter

Tumu Akshar[1]
Department of Computer Science & Information Systems
BITS Pilani Hyderabad Campus
f20200003@hyderabad.bits-pilani.ac.in

Lov Kumar[2]
Department of Computer Engineering
National Institute of Technology, Kurukshetra
lovkumar@nitkkr.ac.in

Yogita[3]
Department of Computer Engineering
National Institute of Technology, Kurukshetra
yogita@nitkkr.ac.in

Lalita Bhanu Murthy[4]
Department of Computer Science & Information Systems
BITS Pilani Hyderabad Campus
bhanu@hyderabad.bits-pilani.ac.in

***Abstract*—In recent times, software developers widely use instant messaging and collaboration platforms, as these platforms aid them in exploring new technologies, raising different development-related issues, and seeking solutions from their peers virtually. Gitter is one such platform that has a heavy userbase. It generates a tremendous volume of data, analysis of which is helpful to gain insights about trends in open-source software development and the developers' inclination toward various technologies. Analyzing these trends helps these platforms better cater to the needs of the developers, in turn increasing the usage of these platforms and promoting collaborations between more developers. The classification techniques can be deployed for this purpose. The selection of an apt word embedding for a given dataset of text messages plays a vital role in determining the performance of classification techniques. In the present work, the comparative analysis of nine-word embeddings in combination with seventeen classification techniques with onevsone and onevsrest has been performed on the GitterCom dataset for categorizing text messages into one of the pre-determined classes based on their purpose. Further, two feature selection methods have been applied. The SMOTE technique has been used for handling data imbalance. It resulted in a total of 1836 classification pipelines for analysis. The objective is to analyze their performances to recommend efficient pipelines for the classification task at hand. The experimental results show that word2vect, GLOVE with 300 vector size, and GLOVE with 100 vector size are three top-performing word embeddings having performance values taken across different classification techniques. The models trained using ANOVA features performed similarly to those models trained using all features. Finally, using the SMOTE technique helps models to get a better prediction ability.**

***Index Terms*—Functional Requirements, Non-Functional Requirements, Deep Learning, Data Imbalance Methods, Feature Selection, Classification Techniques, Word Embedding.**

I. Introduction

THE development of modern complex software systems requires a lot of meticulous planning and large teams of software developers and designers. The members of these teams are often geographically distributed across various locations and rely on online communication and collaboration modes. Many open-source projects and software development teams have shifted towards platforms such as Gitter and Slack due to the features and the support they offer for collaboration between software developers worldwide.Gitter has revolutionized team communications and project coordination, especially for distributed software development teams, by providing a user-friendly way of managing and organizing conversations. Gitter also provides public access to user-generated data, and the historic data is accessible indefinitely through chat room logs. Classifying the purpose of the messages in the developer communications on such platforms assists in better organization of messages into categories, making them easy to be retrieved by the users as per their requirements and deriving various insights into the general trends in open-source software development. This classification also helps understand the major reasons why people use these platforms for. Then, the platforms can be updated to fulfill the requirements of the users, which attracts more people to use these platforms, promoting better collaborations, meaning both the organizations running the platforms and the developers benefit from this. Manual classification of this data is not feasible due to the sheer volume of the data available. So there is a need to deploy Machine Learning (ML) techniques to automate the process and minimize the errors in classification.

The objective of the present work is to perform a comparative analysis of the classification pipelines developed using nine different word embeddings, two feature selection techniques, and thirty-four classifiers. We believe this analysis will help establish a strong foundation for future researchers to select the techniques and pipelines resulting in the best predictive ability of the developed models to identify the purpose of messages in developer communications. For this purpose, GitterCom dataset, which contains around 10,000 messages from various channels on Gitter has been adopted.

Each message in it has three labels viz. Purpose, Category, and Sub-Category, which the dataset creators curated manually. The process of creation of classification pipelines started by applying nine different word-embedding techniques to vectorize the data into a numeric form for further analysis. These word-embedding techniques generate an abundance of features, not all of which are influential in the classification task. Hence, two feature selection techniques, namely One-Way ANOVA and PCA are employed to synthesize or extract the most relevant and influential features to optimize the performance of the models. Then to test the predictive ability of the word-embedding and feature-selection techniques, we have used seventeen different variants of classification techniques with one-vs-one and one-vs-rest approaches. The classification techniques are used by considering various ML and Neural Network (NN) classification algorithms and ensemble learning methods in association with them. The predictive ability of these classifiers was validated using the 5-fold cross-validation method. The SMOTE data sampling technique is also applied to combat data imbalance, leading to incorrect and unreliable results. Finally, the performance of the developed models has been compared using evaluation metrics such as Accuracy, Sensitivity, Specificity, and Geometric Mean, which are extracted using the Box-plot Visualization technique and the Friedman Test.

II. Related Work

The proposed research is based on classifying and analyzing the purpose or rationale behind developer communications on modern instant messaging and communication tools. The developers use it for project coordination or technological discussions. This research also performs a comparative analysis of various NLP models that best analyze the communications on such platforms. In this work, we have used the 'GitterCom' dataset introduced by Parra et al. [1], a manually created dataset of 10,000 messages from developer discussions on the Gitter Platform.

The related work is divided into two sub-sections. The first sub-section focuses on the earlier works that have studied developer communications on instant messaging platforms such as Slack or Gitter. The second sub-section is dedicated to the analysis of messages on other platforms such as Q&A platforms (Eg: Stack Overflow), Social Media Platforms (Eg: Twitter), etc. In both sections, we perform a critical comparison of the previous endeavors with our research approach.

A. Instant Messaging and Communication Tools for Software Developers

Earlier research works performed an empirical study of the properties of the communications on Gitter and Slack. While some of these works focused on topics discussed and problem resolution, others introduced certain classes to classify the purpose or rationale of the messages in such communications. The survey confirmed that various classification techniques were used to identify the topic of discussion or the purpose and rationale of the messages. Ehsan et al.[2] performed an empirical study of developer discussions in Gitter to understand the nature of developer discussions, the type of questions developers ask, and the proposed solutions. The authors proposed an approach for the automatic identification of discussion threads in developer chatrooms using hierarchical clustering algorithms and other heuristics. They identified four patterns of responses based on the response length and complexity. The authors also created a taxonomy of resolution types and discussed topics in the chatrooms. The topics discussed were classified into five types - Installation and configuration, Debugging and troubleshooting, Feature requests and enhancements, Code review and feedback, and General discussion.

Sahar et al. work focused on the analysis of how developers discuss issue reports in Gitter chat rooms related to open-source systems [3]. The authors proposed an approach involving clustering algorithms to identify issue report discussions automatically. These discussions were classified into four types based on the number of messages, participants, and duration of the discussion. In our work, we employ various word embedding techniques to assess which technique captures these key patterns and strategies the best, aiding the classification performance. The above two works broadly focus on identifying the topics of discussion and defining a classification nomenclature based on it. Our work focuses on a possible next step of their research, which is to build and compare Machine Learning pipelines that can automatically classify present and incoming messages based on the considered classification nomenclature.

A study with a research methodology similar to ours was done by Parra et al. [4]; their objective was built upon their previously published paper (Parra et al.[1]) in which they introduced the GitterCom dataset for the first time. The three categories defined to classify each message in the developer discussions based on its purpose in this paper (team-wide, personal benefits, community support) were derived from the work of Lin et al. for the Slack platform [5]. The authors also analyzed the performances of nine supervised machine learning and deep learning algorithms, such as SVM, Decision Trees, AdaBoost, LSTM, etc., along with certain data sampling techniques to classify the purpose of a given message as one of the pre-defined categories. A key difference between our work and theirs is the analysis of the benefits of various word embedding techniques in better capturing the contexts and patterns of the messages which aid the classification. We also explore the possible effects of feature selection techniques since they help reduce the complexity of the models while retaining the key features of the data.

The analysis of the topics and discussion and purpose of messages was also undertaken for Stack Overflow. Lin et al. performed an empirical study to understand the purposes for which developers use Slack [5]. Upon surveying 53 software developers, the authors found that the developers self-reported using Slack for various purposes. They broadly classified these purposes into personal benefits, community support, and team-wide purposes. Another such empirical study was undertaken by Stray et al. [6]. The authors explored the use of Slack for communication and project coordination in virtual agile development teams as they studied the communications of a group of 30 software developers at a large software development organization. They found that the messages could be broadly classified into one of the following purposes: general information/coordination, general discussions, problem-focused communication, technical communication, and socializing.

Alkadhi et al. performed an exploratory study to investigate the presence of rationales in chat messages during software development [7]. They collected and analyzed chat messages from three open-source software development teams on GitHub comprising students working on capstone projects. Based on this analysis, the authors defined a set of categories for the types of rationale found in the messages - Design, Functionality, Implementation, and Maintenance. Their findings also indicate the usefulness of classification algorithms such as SVM and Naïve Bayes and the data sampling techniques for automatically identifying and classifying messages based on the rational information they contain. Our work, however, focused on recommending the best techniques and pipelines for building such models to extract the best performance out of them rather than just checking for feasibility. In subsequent work, Alkadhi et al. presented a new approach called REACT (Rationale Extraction from Chat Transcripts) [8]. They considered five rationale elements for their work - issues, alternatives, pro-arguments, con-arguments, and decisions. The REACT approach enabled the developers to explicitly record the rationale in messages on the Slack platform through manual annotation.

B. Other Communication Tools for Software Developers

Various other platforms facilitate software development communication. The concept of such tools is based on Question & Answer forums like Stack Overflow and social media platforms like Twitter. Software developers spread out across various geographic locations relied majorly on such platforms for communication and coordination before the advent of the modern instant messaging tools dedicated to facilitating such discussions. Q&A platforms such as Stack Overflow encourage efficient problem-solving, community support, and knowledge sharing among its users across the globe. Social Media sites like Twitter also help in this regard by providing real-time updates, networking opportunities, and crowd-sourced solutions through hashtags and mentions to reach out to relevant experts for assistance.

The current tags provided for questions on Stack Overflow are based on the technologies used or being referred to in the question. Classifying these questions based on the purpose or context will serve the users better since it makes the process of finding relevant posts quicker. Beyer et al. performed such work where they obtained a taxonomy of seven question categories: API change, API usage, Conceptual, Discrepancy, Learning, Errors, and Review, and they manually curated a dataset consisting of 500 posts classified into these categories [9]. The authors then developed classification models using the Random Forest and SVM algorithms along with data sampling techniques and performed a comparative analysis with 82 different configurations regarding the preprocessing and representation of the input text data to analyze which configuration captured the context and intricacies of the data better. In place of such configurations, we explored word embeddings for this reason since they offer better generalizability to out-of-vocabulary words as they provide continuous representations of words by considering semantic equivalence, something which pre-processing techniques might struggle with.

Venigalla et al. undertook a similar study using the Latent Dirichlet Allocation Algorithm to model six questions topics based on their purpose [10]. The authors also came up with names for those topics based on the taxonomy used in the literature. They also used various Machine Learning algorithms, such as Linear SVC, Logistic Regression, Multinomial Naive Bayes, Random Forest, etc., to develop classification models for the questions on Stack Overflow.

Guzman et al. performed an analysis of the tweets on Twitter related to software applications[11]. The intention was to obtain tweets that could be useful for developers by using classification algorithms. The purpose of these tweets includes improvement suggestions, user needs, bug reports, feature requests, etc. The authors introduced ALERTme, an approach to automatically classify, group, and rank such tweets. This approach relied on classification algorithms such as Multinomial Naive Bayes and word-embedding techniques such as TF-IDF for classification.

We preferred Gitter over such platforms because Gitter's data has been largely untapped for such analysis, and Gitter provides more intricate and essential details of software development pipelines and requirements as developers use it regularly to discuss such details.

III. STUDY DESIGN

This section presents the details regarding various design setting used for this research.

A. Experimental Dataset

We plan to use the GitterCom dataset for this experiment, the first manually labeled dataset of Gitter instant message histories in open-source systems. It contains 10,000 messages collected from 10 open-source software development Gitter communities (1,000 messages per community). Each message in this dataset was manually labeled to capture the purpose of the communication expressed by the message. Each record in the dataset consists of the following information: (i) The channel of communication (ii) the Message ID (iii) The date and time when the message was posted (iv) The author of the message (v) The message in plain text (vi) Purpose of the message (vii) The subclass of the purpose it belongs to - category (viii) The subclass of the category it belongs to i.e., the sub-category.

In this experiment, we intend to predict the Purpose label for any given message. There are three possible classes (labels) of purpose:

- **Personal Benefits** - Includes messages posted to satisfy the developer's personal needs.
- **Team-wide activities** - Includes messages related to the discussion among the members of a team working on software development activities related to the system they are developing.
- **Community Support** - Includes messages posted for general discussions among developers from communities or special-interest groups who intend to explore new tools and frameworks or brainstorm ideas.

B. Word Embeddings

The textual data of the dataset is to be represented in numeric vector form for further analysis. To achieve this, nine different word embedding techniques, including Continuous Bag of Words (CBOW), Skip-Gram (SKG), Global Vectors for Word Representation (GloVe) (with 50 dimensions, 100 dimensions, and 300 dimensions), Word2vec, fastText (FST), Generative Pre-trained Transformer (GPT), and Generative Pre-trained Transformer-2 (GPT-2), will be applied on the dataset. These techniques help represent the textual data as a vector in an n-dimensional space. All the stopwords, spaces, and other bad symbols will be removed from the textual data before applying the word embedding techniques. The generated vectorial representations will then be used to develop models to determine a message's purpose. [12].

C. Feature Selection Techniques

Since vectorial representations are used as inputs to develop the classification models, and each of the vectorial representations contains a lot of columns (features), some of which may not be as influential as the others, optimization of the data and selection of the influential features becomes crucial to improve the performance of the models. To extract the important features, we plan to use two different Feature Selection Techniques - One-Way Analysis of Variance (One-Way ANOVA) and Principal Component Analysis (PCA) to discard irrelevant features and obtain the set of relevant and influential features.

D. Training of Models from Imbalanced Data Set

Upon analysis of the dataset, if we observe the problem of class imbalance in our dataset, i.e., the number of data samples in each class is different, we resort to the data sampling techniques to enhance the performance of the developed models. We propose to use the Minority Oversampling Technique (SMOTE) on the dataset to balance the data in such a scenario.

E. Classification Techniques

We propose to develop thirty-four different classifiers to perform a comparative analysis of the developed models. For that, classification algorithms such as Multinomial Naive Bayes (MNB), Bernoulli's Naive Bayes (BNB), Gaussian Naive Bayes (GNB), Decision Tree Classifier, Logistic Regression, K-Nearest Neighbours Classifier, Random Forest Classifier, Extra Trees Classifier, Multi-Layer Perceptron with Limited-memory BFGS, Stochastic Gradient Descent, and Adam Optimizers will be utilized along with ensemble modeling techniques such as Bagging with KNN, Multinomial Naive Bayes, Logistic Regression and Decision Tree Classifiers, Ada Boost Classifier and Gradient Boosting Classifier. Each classification technique mentioned above will be implemented using one-vs-Rest and One-vs-One Multi-class classification strategies, thus giving a total of thirty-four classifiers. These classification strategies are defined below:

- **One-vs-One classification**: The model trains on pair-wise comparisons between each possible combination of classes. The final prediction is made by aggregating the votes from all binary classifiers.
- **One-vs-Rest classification**: The model trains a binary classifier for each class to distinguish between that class and the rest of the classes combined. The item is assigned to the class having the maximum probability of having that item.

IV. Research Methodology

In this work, we started with pre-processing the dataset by removing unnecessary punctuation, spaces, and stop-words. We also had to manually delete some records that consisted of messages with empty or unrecognizable symbols. After obtaining the pre-processed dataset, We applied nine different word embedding techniques to extract numeric feature vectors from the messages on the Gitter platform. We considered these features as inputs to develop models for predicting the purpose label of a given message. After applying the word-embedding techniques, we employed the One-Way ANOVA, and PCA to obtain the best combination of relevant features. The One-Way ANOVA test helps find features with equal variance between groups, which means these features don't impact the response much and hence can be discarded. PCA considerably reduces the number of features while trying to retain a significant portion of the variance in the original dataset.

To make the computation of the models simpler and feasible, we sampled 3000 rows randomly from the dataset. Then, we observed the presence of a class imbalance in the sampled rows as the no. of messages with the 'Team-wide' purpose label was much higher than other labels. To combat this issue, we employed the Synthetic Minority Over-sampling (SMOTE) technique, which creates synthetic data samples based on the original data. The models obtained using the above techniques were trained using 34 different classifiers developed using various classification algorithms and ensemble methods and implemented using both One vs. One and One vs. Rest Classification strategies. The predictive ability of these classifiers was validated using the 5-fold cross-validation method. The performance of the developed models was compared using evaluation metrics such as Accuracy, Sensitivity, Specificity, and Geometric Mean, which were extracted using the Box-plot Visualization technique and the Friedman Test.

The detailed overview of the proposed framework is shown in Figure 1. The initial glance of the figure suggests that the

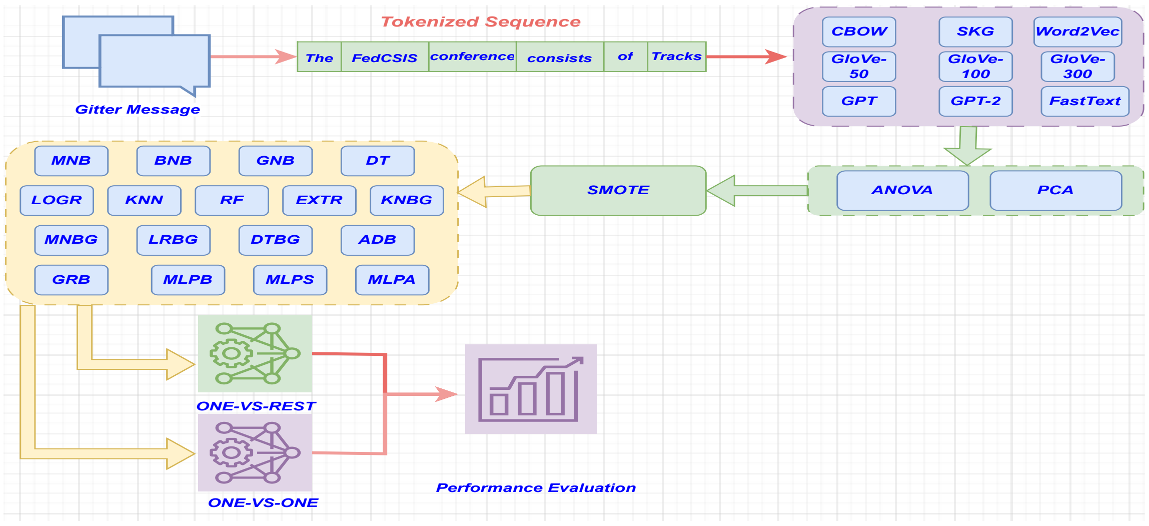

Fig. 1: Framework of proposed work

proposed framework is a multi-step process involving feature extraction from text data using word embedding techniques, removal of irrelevant features, handling class imbalance using SMOTE and development of prediction model using seventeen classification algorithms implemented using two multi-class classification strategies. First, the messages in the GitterCom dataset is pre-processed by removing unnecessary characters and stopwords. Then as shown in Figure 1, these messages are tokenized and nine different word embedding techniques are applied to find the numeric vector representations of these messages. Then the feature selection techniques of One-Way ANOVA and PCA are employed to discard the non-influential features of the numeric data. The next step involves applying the SMOTE technique to handle the presence of class imbalance in the dataset.Then, the prediction models are developed using seventeen classification algorithms, implemented using two multi-class classification strategies – One vs One and One vs Rest. The performance of the models developed using these two strategies is then evaluated using various evaluation metrics.

V. Empirical Results and Analysis

In this work, we applied nine different word embedding techniques, two feature selection techniques, one class balancing technique, and thirty-four different classifiers to analyze and classify the purpose of the messages posted on the Gitter platform. Thus, a total of 1836 [1 dataset * 9 word-embedding techniques * (2 sets generated using feature selection techniques + 1 set of original data) * (1 set generated using a class balancing technique + 1 original dataset) * 34 classifiers] distinct predictive models were built. The predictive ability of the developed models was evaluated using the Accuracy, Sensitivity, Specificity, and Geometric Mean (G-Mean) metrics. These models were validated using the 5-fold cross-validation method.

Accuracy is the most commonly used metric for evaluating the performance of a classifier. It is a measure of the proportion of the correctly classified instances out of the total number of available instances. While it is a useful metric, it can give misleading results in the presence of a class imbalance in the data. Hence, we also incorporate the Sensitivity and Specificity metrics to deal with this issue. While sensitivity is the measure of the proportion of the actual positive cases correctly identified by the classifier, Specificity is the measure of the proportion of the actual negative cases correctly identified by the classifier. Although these two metrics give better predictions than Accuracy in the presence of imbalanced data, there are some instances where these metrics fail, such as when the classifier always predicts the majority class; the models would get high sensitivity but low Specificity. To deal with such cases, we also consider the Geometric Mean metric, which is the geometric mean of both Sensitivity and Specificity.

Tables I and II report the accuracies obtained for the various models developed by applying different classifiers to original data and sampled data on different sets of features. By analyzing the information in the table, the following inferences can be derived that the model with the highest accuracy is obtained by applying the Random Forest Classifier using the One vs Rest classification strategy on the data obtained from the Word2Vec word embedding technique and employing the SMOTE technique to address the class imbalance problem. Similarly, Table I reports the Geometric Mean (G-Mean) values obtained for the various models developed by applying different classifiers to original data and sampled data on different sets of features. By analyzing the information in the table, the following inferences can be derived that the model with the highest accuracy is obtained by applying the Random Forest Classifier using the One vs Rest classification strategy on the data obtained from the Word2Vec word embedding technique and employing the SMOTE technique to address the class

TABLE I: Accuracy and G-Mean

Accuracy																		
	ORG_DATA									**SMOTE_DATA**								
Embedding	**MNB**	**DTC**	**LRC**	**MNBG**	**LRBG**	**DTBG**	**RF**	**ADB**	**MLPB**	**MNB**	**DTC**	**LRC**	**MNBG**	**LRBG**	**DTBG**	**RF**	**ADB**	**MLPB**
One-Vs-One: AF																		
CBOW	96.97	90.10	96.97	96.97	96.97	96.87	96.77	96.93	96.43	60.86	93.21	69.41	60.55	67.68	97.01	98.04	80.92	52.07
SKG	96.77	90.67	96.97	96.97	96.97	96.97	96.93	96.80	95.70	63.07	93.69	80.72	62.79	76.06	97.83	98.54	84.06	98.72
GLOVE50	96.97	90.47	96.97	96.97	96.97	97.00	96.93	96.90	95.40	65.44	92.38	77.69	64.70	71.43	98.22	98.83	80.81	32.38
GLOVE100	96.97	89.83	96.97	96.97	96.97	96.93	96.93	96.73	95.70	64.65	91.99	88.01	64.83	81.41	98.68	98.99	82.31	87.65
GLOVE300	96.90	90.93	97.00	96.97	96.97	96.97	96.97	96.80	95.20	71.07	93.35	95.66	70.98	92.88	98.88	99.56	88.04	97.17
W2V	96.97	90.60	96.93	96.97	96.97	96.93	96.93	96.73	95.80	72.63	94.13	96.51	72.52	94.50	99.37	99.70	88.77	63.78
FST	96.67	90.27	96.97	96.97	96.97	96.87	97.00	96.67	95.27	61.98	94.00	76.82	60.62	73.79	97.79	98.58	84.04	91.84
GPT	92.93	95.43	96.97	94.60	96.97	96.83	96.53	96.90	96.97	46.11	94.86	81.24	46.01	75.42	96.10	96.24	79.34	95.36
GPT2	95.63	95.90	96.93	95.77	96.97	96.87	96.60	96.83	96.97	55.53	94.17	90.52	56.10	89.50	96.31	96.53	83.87	94.04
One-Vs-One: ANOVA																		
CBOW	96.97	90.67	96.97	96.97	96.97	96.77	96.83	96.93	96.93	60.77	92.96	69.51	60.32	67.61	97.00	97.88	80.18	91.93
SKG	96.93	91.13	96.97	96.97	96.97	96.93	96.90	96.93	96.07	63.56	93.92	80.07	62.91	75.67	98.01	98.52	84.24	95.89
GLOVE50	96.97	90.40	96.97	96.97	96.97	96.93	96.97	96.83	96.10	65.53	92.37	76.93	61.97	71.31	98.17	98.92	79.08	32.55
GLOVE100	96.97	90.03	96.97	96.97	96.97	96.93	96.93	96.83	95.33	64.67	91.98	86.67	65.36	80.76	98.40	99.16	80.84	96.31
GLOVE300	96.97	92.10	96.97	96.97	96.97	96.93	96.93	96.83	95.67	70.95	93.51	95.37	70.69	92.24	98.97	99.34	87.90	98.03
W2V	96.97	90.97	96.97	96.97	96.97	96.83	96.83	96.70	96.97	72.40	94.02	96.15	72.05	93.64	99.30	99.62	88.89	32.24
FST	96.77	90.60	96.97	96.97	96.97	96.83	96.80	96.60	95.87	61.71	93.62	76.83	61.36	72.77	97.88	98.53	82.30	98.25
GPT	93.17	95.43	96.97	94.67	96.97	96.87	96.50	96.93	96.97	47.90	94.47	79.03	47.12	73.83	96.07	96.46	79.16	49.48
GPT2	95.67	95.70	96.97	95.77	96.97	96.90	96.70	96.97	96.60	55.67	94.29	90.28	55.56	89.47	96.21	96.29	84.18	94.80
One-Vs-Rest: AF																		
CBOW	96.97	93.77	96.97	96.97	96.97	96.70	96.77	96.93	95.30	63.68	94.98	68.81	62.68	67.02	96.96	97.90	80.75	31.91
SKG	96.73	93.13	96.97	96.97	96.97	96.93	96.90	96.77	95.43	63.76	95.06	79.58	63.29	74.99	97.82	98.59	84.51	77.33
GLOVE50	96.97	93.40	96.97	96.97	96.97	96.97	96.97	96.60	94.67	65.89	94.36	79.13	64.64	72.90	97.97	98.68	80.66	32.38
GLOVE100	96.97	92.87	96.97	96.97	96.97	96.97	96.93	96.70	92.93	66.88	95.36	87.40	67.09	81.29	98.38	99.13	81.22	97.54
GLOVE300	96.47	93.13	97.00	96.97	96.97	96.93	96.93	96.40	94.50	73.92	95.39	96.07	73.89	93.22	98.73	99.43	86.51	46.02
W2V	96.97	93.27	96.93	96.97	96.97	96.97	96.93	96.47	93.73	74.60	96.38	96.92	74.55	95.10	99.06	99.53	89.19	52.53
FST	96.57	93.33	96.97	96.97	96.97	96.87	96.83	96.60	94.80	63.08	95.28	77.36	62.82	73.85	97.74	98.42	82.98	52.65
GPT	89.33	96.03	96.97	94.47	96.97	96.90	96.57	96.87	96.97	48.38	94.96	81.15	47.52	76.91	95.76	96.22	79.66	93.97
GPT2	95.50	96.23	96.90	95.77	96.97	96.93	96.73	96.83	96.97	56.14	95.27	90.60	55.60	89.68	96.04	96.39	84.36	95.94
One-Vs-Rest: ANOVA																		
CBOW	96.97	93.37	96.97	96.97	96.97	96.87	96.73	96.70	96.97	63.74	95.08	68.96	63.22	67.11	96.57	97.67	81.17	88.24
SKG	96.70	93.57	96.97	96.97	96.97	96.97	96.83	96.70	95.33	63.70	95.44	78.72	62.39	74.55	97.82	98.40	83.37	32.51
GLOVE50	96.97	93.73	96.97	96.97	96.97	96.93	96.93	96.57	95.27	65.67	94.76	77.80	64.76	71.62	98.04	98.87	80.12	32.55
GLOVE100	96.97	93.57	96.97	96.97	96.97	96.93	96.93	96.77	94.57	67.16	95.46	86.64	67.40	80.27	98.38	98.89	80.61	97.23
GLOVE300	96.40	93.57	96.97	96.97	96.97	96.87	96.93	96.27	94.13	73.35	95.86	95.61	72.95	92.49	98.79	99.43	86.97	58.38
W2V	96.97	93.67	96.93	96.97	96.97	96.97	96.93	96.37	93.57	74.36	96.00	96.52	74.18	94.34	99.13	99.63	88.75	32.24
FST	96.67	93.43	96.97	96.97	96.97	96.83	96.90	96.43	94.30	62.97	95.51	77.32	62.77	73.24	97.90	98.66	82.40	89.85
GPT	89.43	96.07	96.97	94.27	96.97	96.83	96.50	96.93	96.97	46.30	94.81	80.17	46.68	75.68	95.82	96.16	78.66	40.98
GPT2	95.50	96.30	96.97	95.77	96.97	96.70	96.77	96.87	96.33	54.97	94.90	90.50	55.30	89.60	95.89	96.28	84.67	91.68
G-Mean																		
One-Vs-One: AF																		
CBOW	0.17	0.42	0.17	0.17	0.17	0.20	0.17	0.17	0.17	0.70	0.95	0.77	0.70	0.75	0.98	0.99	0.86	0.63
SKG	0.17	0.38	0.17	0.17	0.17	0.20	0.22	0.20	0.33	0.72	0.95	0.85	0.71	0.82	0.98	0.99	0.88	0.99
GLOVE50	0.17	0.31	0.17	0.17	0.17	0.20	0.17	0.20	0.26	0.74	0.94	0.83	0.73	0.78	0.99	0.99	0.85	0.46
GLOVE100	0.17	0.32	0.17	0.17	0.17	0.17	0.17	0.22	0.28	0.73	0.94	0.91	0.73	0.86	0.99	0.99	0.87	0.91
GLOVE300	0.17	0.32	0.20	0.17	0.17	0.17	0.20	0.20	0.35	0.78	0.95	0.97	0.78	0.95	0.99	1.00	0.91	0.98
W2V	0.17	0.32	0.17	0.17	0.17	0.17	0.17	0.20	0.28	0.79	0.96	0.97	0.79	0.96	1.00	1.00	0.92	0.72
FST	0.22	0.35	0.17	0.17	0.17	0.20	0.22	0.22	0.32	0.71	0.95	0.82	0.70	0.80	0.98	0.99	0.88	0.94
GPT	0.43	0.24	0.17	0.42	0.17	0.17	0.20	0.17	0.17	0.58	0.96	0.86	0.58	0.81	0.97	0.97	0.84	0.97
GPT2	0.43	0.26	0.17	0.43	0.17	0.17	0.25	0.17	0.17	0.66	0.96	0.93	0.66	0.92	0.97	0.97	0.88	0.96
One-Vs-One: ANOVA																		
CBOW	0.17	0.40	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.70	0.95	0.77	0.70	0.75	0.98	0.98	0.85	0.94
SKG	0.17	0.46	0.17	0.17	0.17	0.17	0.22	0.20	0.20	0.72	0.95	0.85	0.72	0.82	0.99	0.99	0.88	0.97
GLOVE50	0.17	0.32	0.17	0.17	0.17	0.17	0.20	0.17	0.17	0.74	0.94	0.82	0.71	0.78	0.99	0.99	0.84	0.46
GLOVE100	0.17	0.29	0.17	0.17	0.17	0.17	0.17	0.20	0.24	0.73	0.94	0.90	0.74	0.85	0.99	0.99	0.85	0.97
GLOVE300	0.17	0.37	0.17	0.17	0.17	0.17	0.17	0.22	0.30	0.78	0.95	0.97	0.78	0.94	0.99	1.00	0.91	0.99
W2V	0.17	0.34	0.17	0.17	0.17	0.17	0.17	0.22	0.17	0.79	0.96	0.97	0.79	0.95	0.99	1.00	0.92	0.46
FST	0.27	0.43	0.17	0.17	0.17	0.20	0.20	0.28	0.33	0.71	0.95	0.82	0.70	0.79	0.98	0.99	0.87	0.99
GPT	0.43	0.22	0.17	0.42	0.17	0.17	0.22	0.17	0.17	0.60	0.96	0.84	0.59	0.80	0.97	0.97	0.84	0.61
GPT2	0.43	0.24	0.17	0.43	0.17	0.17	0.20	0.17	0.17	0.66	0.96	0.93	0.66	0.92	0.97	0.97	0.88	0.96
One-Vs-Rest: AF																		
CBOW	0.17	0.36	0.17	0.17	0.17	0.17	0.22	0.17	0.20	0.72	0.96	0.76	0.71	0.75	0.98	0.98	0.85	0.46
SKG	0.25	0.34	0.17	0.17	0.17	0.22	0.22	0.22	0.32	0.72	0.96	0.85	0.72	0.81	0.98	0.99	0.88	0.83
GLOVE50	0.17	0.19	0.17	0.17	0.17	0.20	0.20	0.25	0.31	0.74	0.96	0.84	0.73	0.79	0.98	0.99	0.85	0.46
GLOVE100	0.17	0.22	0.17	0.17	0.17	0.20	0.17	0.22	0.29	0.75	0.97	0.90	0.75	0.86	0.99	0.99	0.86	0.98
GLOVE300	0.27	0.30	0.20	0.17	0.17	0.17	0.20	0.24	0.34	0.80	0.97	0.97	0.80	0.95	0.99	1.00	0.90	0.58
W2V	0.17	0.26	0.17	0.17	0.17	0.17	0.17	0.27	0.40	0.81	0.97	0.98	0.81	0.96	0.99	1.00	0.92	0.63
FST	0.27	0.28	0.17	0.17	0.17	0.17	0.25	0.22	0.35	0.72	0.96	0.83	0.72	0.80	0.98	0.99	0.87	0.63
GPT	0.44	0.20	0.17	0.42	0.17	0.17	0.17	0.17	0.17	0.60	0.96	0.86	0.59	0.82	0.97	0.97	0.85	0.95
GPT2	0.43	0.24	0.17	0.43	0.17	0.17	0.25	0.17	0.17	0.66	0.96	0.93	0.66	0.92	0.97	0.97	0.88	0.97
One-Vs-Rest: ANOVA																		
CBOW	0.17	0.30	0.17	0.17	0.17	0.17	0.17	0.17	0.17	0.72	0.96	0.76	0.72	0.75	0.97	0.98	0.86	0.91
SKG	0.27	0.37	0.17	0.17	0.17	0.22	0.20	0.28	0.33	0.72	0.97	0.84	0.71	0.81	0.98	0.99	0.87	0.46
GLOVE50	0.17	0.26	0.17	0.17	0.17	0.17	0.17	0.20	0.24	0.74	0.96	0.83	0.73	0.78	0.99	0.99	0.85	0.46
GLOVE100	0.17	0.28	0.17	0.17	0.17	0.17	0.17	0.20	0.28	0.75	0.97	0.90	0.75	0.85	0.99	0.99	0.85	0.98
GLOVE300	0.22	0.24	0.17	0.17	0.17	0.17	0.17	0.30	0.34	0.80	0.97	0.97	0.79	0.94	0.99	1.00	0.90	0.68
W2V	0.17	0.22	0.17	0.17	0.17	0.17	0.17	0.28	0.40	0.81	0.97	0.97	0.80	0.96	0.99	1.00	0.92	0.46
FST	0.28	0.30	0.17	0.17	0.17	0.20	0.22	0.30	0.34	0.72	0.97	0.83	0.71	0.80	0.98	0.99	0.87	0.92
GPT	0.47	0.17	0.17	0.42	0.17	0.17	0.20	0.17	0.17	0.58	0.96	0.85	0.59	0.82	0.97	0.97	0.84	0.54
GPT2	0.43	0.17	0.17	0.43	0.17	0.17	0.17	0.17	0.20	0.65	0.96	0.93	0.66	0.92	0.97	0.97	0.88	0.94

imbalance problem. This model gives the highest classification accuracy, sensitivity, and specificity, as mentioned in Table II.

A. *COMPARATIVE ANALYSIS*

In this section, we examine and compare the efficacy of the models created through a diverse set of word-embedding

TABLE II: Sensitivity and Specificity

Sensitivity																		
	ORG_DATA									SMOTE_DATA								
Embedding	**MNB**	**DTC**	**LRC**	**MNBG**	**LRBG**	**DTBG**	**RF**	**ADB**	**MLPB**	**MNB**	**DTC**	**LRC**	**MNBG**	**LRBG**	**DTBG**	**RF**	**ADB**	**MLPB**
One-Vs-One: AF																		
CBOW	0.97	0.90	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.20	0.03	0.03	0.03	0.04	0.03	0.03	0.03
SKG	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.16	0.03	0.03	0.03	0.04	0.05	0.04	0.12
GLOVE50	0.97	0.90	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.10	0.03	0.03	0.03	0.04	0.03	0.04	0.07
GLOVE100	0.97	0.90	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.11	0.03	0.03	0.03	0.03	0.03	0.05	0.08
GLOVE300	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.11	0.04	0.03	0.03	0.03	0.04	0.04	0.13
W2V	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.11	0.03	0.03	0.03	0.03	0.03	0.04	0.08
FST	0.97	0.90	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.05	0.14	0.03	0.03	0.03	0.04	0.05	0.05	0.10
GPT	0.93	0.95	0.97	0.95	0.97	0.97	0.97	0.97	0.97	0.20	0.06	0.03	0.19	0.03	0.03	0.04	0.03	0.03
GPT2	0.96	0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.19	0.07	0.03	0.19	0.03	0.03	0.06	0.03	0.03
One-Vs-One: ANOVA																		
CBOW	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.03	0.18	0.03	0.03	0.03	0.03	0.03	0.03	0.03
SKG	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.23	0.03	0.03	0.03	0.03	0.05	0.04	0.04
GLOVE50	0.97	0.90	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.11	0.03	0.03	0.03	0.03	0.04	0.03	0.03
GLOVE100	0.97	0.90	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.09	0.03	0.03	0.03	0.03	0.03	0.04	0.06
GLOVE300	0.97	0.92	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.03	0.15	0.03	0.03	0.03	0.03	0.03	0.05	0.09
W2V	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.03	0.13	0.03	0.03	0.03	0.03	0.03	0.05	0.03
FST	0.97	0.91	0.97	0.97	0.97	0.97	0.97	0.97	0.96	0.07	0.20	0.03	0.03	0.03	0.04	0.04	0.08	0.12
GPT	0.93	0.95	0.97	0.95	0.97	0.97	0.97	0.97	0.97	0.20	0.05	0.03	0.19	0.03	0.03	0.05	0.03	0.03
GPT2	0.96	0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.19	0.06	0.03	0.19	0.03	0.03	0.04	0.03	0.03
One-Vs-Rest: AF																		
CBOW	0.61	0.93	0.69	0.61	0.68	0.97	0.98	0.81	0.52	0.80	0.97	0.85	0.80	0.84	0.99	0.99	0.90	0.76
SKG	0.63	0.94	0.81	0.63	0.76	0.98	0.99	0.84	0.99	0.82	0.97	0.90	0.81	0.88	0.99	0.99	0.92	0.99
GLOVE50	0.65	0.92	0.78	0.65	0.71	0.98	0.99	0.81	0.32	0.83	0.96	0.89	0.82	0.86	0.99	0.99	0.90	0.66
GLOVE100	0.65	0.92	0.88	0.65	0.81	0.99	0.99	0.82	0.88	0.82	0.96	0.94	0.82	0.91	0.99	0.99	0.91	0.94
GLOVE300	0.71	0.93	0.96	0.71	0.93	0.99	1.00	0.88	0.97	0.86	0.97	0.98	0.85	0.96	0.99	1.00	0.94	0.99
W2V	0.73	0.94	0.97	0.73	0.94	0.99	1.00	0.89	0.64	0.86	0.97	0.98	0.86	0.97	1.00	1.00	0.94	0.82
FST	0.62	0.94	0.77	0.61	0.74	0.98	0.99	0.84	0.92	0.81	0.97	0.88	0.80	0.87	0.99	0.99	0.92	0.96
GPT	0.46	0.95	0.81	0.46	0.75	0.96	0.96	0.79	0.95	0.73	0.97	0.91	0.73	0.88	0.98	0.98	0.90	0.98
GPT2	0.56	0.94	0.91	0.56	0.90	0.96	0.97	0.84	0.94	0.78	0.97	0.95	0.78	0.95	0.98	0.98	0.92	0.97
One-Vs-Rest: ANOVA																		
CBOW	0.61	0.93	0.70	0.60	0.68	0.97	0.98	0.80	0.92	0.80	0.96	0.85	0.80	0.84	0.98	0.99	0.90	0.96
SKG	0.64	0.94	0.80	0.63	0.76	0.98	0.99	0.84	0.96	0.82	0.97	0.90	0.81	0.88	0.99	0.99	0.92	0.98
GLOVE50	0.66	0.92	0.77	0.62	0.71	0.98	0.99	0.79	0.33	0.83	0.96	0.88	0.81	0.86	0.99	0.99	0.90	0.66
GLOVE100	0.65	0.92	0.87	0.65	0.81	0.98	0.99	0.81	0.96	0.82	0.96	0.93	0.83	0.90	0.99	1.00	0.90	0.98
GLOVE300	0.71	0.94	0.95	0.71	0.92	0.99	0.99	0.88	0.98	0.85	0.97	0.98	0.85	0.96	0.99	1.00	0.94	0.99
W2V	0.72	0.94	0.96	0.72	0.94	0.99	1.00	0.89	0.32	0.86	0.97	0.98	0.86	0.97	1.00	1.00	0.94	0.66
FST	0.62	0.94	0.77	0.61	0.73	0.98	0.99	0.82	0.98	0.81	0.97	0.88	0.81	0.86	0.99	0.99	0.91	0.99
GPT	0.48	0.94	0.79	0.47	0.74	0.96	0.96	0.79	0.49	0.74	0.97	0.90	0.74	0.87	0.98	0.98	0.90	0.75
GPT2	0.56	0.94	0.90	0.56	0.89	0.96	0.96	0.84	0.95	0.78	0.97	0.95	0.78	0.95	0.98	0.98	0.92	0.97
Specificity																		
One-Vs-One: AF																		
CBOW	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.14	0.03	0.03	0.03	0.03	0.05	0.03	0.04
SKG	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.06	0.13	0.03	0.03	0.03	0.05	0.05	0.05	0.10
GLOVE50	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.04	0.03	0.03	0.03	0.04	0.04	0.06	0.10
GLOVE100	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.93	0.03	0.05	0.03	0.03	0.03	0.04	0.03	0.05	0.09
GLOVE300	0.96	0.93	0.97	0.97	0.97	0.97	0.97	0.96	0.95	0.07	0.09	0.04	0.03	0.03	0.03	0.04	0.06	0.13
W2V	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.96	0.94	0.03	0.07	0.03	0.03	0.03	0.03	0.03	0.07	0.17
FST	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.07	0.08	0.03	0.03	0.03	0.03	0.06	0.05	0.13
GPT	0.89	0.96	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.22	0.04	0.03	0.19	0.03	0.03	0.03	0.03	0.03
GPT2	0.96	0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.19	0.06	0.03	0.19	0.03	0.03	0.06	0.03	0.03
One-Vs-One: ANOVA																		
CBOW	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.03	0.09	0.03	0.03	0.03	0.03	0.03	0.03	0.03
SKG	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.07	0.15	0.03	0.03	0.03	0.05	0.04	0.08	0.12
GLOVE50	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.07	0.03	0.03	0.03	0.03	0.03	0.04	0.06
GLOVE100	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.97	0.95	0.03	0.08	0.03	0.03	0.03	0.03	0.03	0.04	0.08
GLOVE300	0.96	0.94	0.97	0.97	0.97	0.97	0.97	0.96	0.94	0.05	0.06	0.03	0.03	0.03	0.03	0.03	0.09	0.13
W2V	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.96	0.94	0.03	0.05	0.03	0.03	0.03	0.03	0.03	0.08	0.17
FST	0.97	0.93	0.97	0.97	0.97	0.97	0.97	0.96	0.94	0.08	0.09	0.03	0.03	0.03	0.04	0.05	0.09	0.13
GPT	0.89	0.96	0.97	0.94	0.97	0.97	0.97	0.97	0.97	0.24	0.03	0.03	0.19	0.03	0.03	0.04	0.03	0.03
GPT2	0.96	0.96	0.97	0.96	0.97	0.97	0.97	0.97	0.96	0.19	0.03	0.03	0.19	0.03	0.03	0.03	0.03	0.04
One-Vs-Rest: AF																		
CBOW	0.64	0.95	0.69	0.63	0.67	0.97	0.98	0.81	0.32	0.82	0.97	0.84	0.81	0.84	0.98	0.99	0.90	0.66
SKG	0.64	0.95	0.80	0.63	0.75	0.98	0.99	0.85	0.77	0.82	0.98	0.90	0.82	0.87	0.99	0.99	0.92	0.89
GLOVE50	0.66	0.94	0.79	0.65	0.73	0.98	0.99	0.81	0.32	0.83	0.97	0.90	0.82	0.86	0.99	0.99	0.90	0.66
GLOVE100	0.67	0.95	0.87	0.67	0.81	0.98	0.99	0.81	0.98	0.83	0.98	0.94	0.84	0.91	0.99	1.00	0.91	0.99
GLOVE300	0.74	0.95	0.96	0.74	0.93	0.99	0.99	0.87	0.46	0.87	0.98	0.98	0.87	0.97	0.99	1.00	0.93	0.73
W2V	0.75	0.96	0.97	0.75	0.95	0.99	1.00	0.89	0.53	0.87	0.98	0.98	0.87	0.98	1.00	1.00	0.95	0.76
FST	0.63	0.95	0.77	0.63	0.74	0.98	0.98	0.83	0.53	0.82	0.98	0.89	0.81	0.87	0.99	0.99	0.91	0.76
GPT	0.48	0.95	0.81	0.48	0.77	0.96	0.96	0.80	0.94	0.74	0.97	0.91	0.74	0.88	0.98	0.98	0.90	0.97
GPT2	0.56	0.95	0.91	0.56	0.90	0.96	0.96	0.84	0.96	0.78	0.98	0.95	0.78	0.95	0.98	0.98	0.92	0.98
One-Vs-Rest: ANOVA																		
CBOW	0.64	0.95	0.69	0.63	0.67	0.97	0.98	0.81	0.88	0.82	0.98	0.84	0.82	0.84	0.98	0.99	0.91	0.94
SKG	0.64	0.95	0.79	0.62	0.75	0.98	0.98	0.83	0.33	0.82	0.98	0.89	0.81	0.87	0.99	0.99	0.92	0.66
GLOVE50	0.66	0.95	0.78	0.65	0.72	0.98	0.99	0.80	0.33	0.83	0.97	0.89	0.82	0.86	0.99	0.99	0.90	0.66
GLOVE100	0.67	0.95	0.87	0.67	0.80	0.98	0.99	0.81	0.97	0.84	0.98	0.93	0.84	0.90	0.99	0.99	0.90	0.99
GLOVE300	0.73	0.96	0.96	0.73	0.92	0.99	0.99	0.87	0.58	0.87	0.98	0.98	0.86	0.96	0.99	1.00	0.93	0.79
W2V	0.74	0.96	0.97	0.74	0.94	0.99	1.00	0.89	0.32	0.87	0.98	0.98	0.87	0.97	1.00	1.00	0.94	0.66
FST	0.63	0.96	0.77	0.63	0.73	0.98	0.99	0.82	0.90	0.81	0.98	0.89	0.81	0.87	0.99	0.99	0.91	0.95
GPT	0.46	0.95	0.80	0.47	0.76	0.96	0.96	0.79	0.41	0.73	0.97	0.90	0.73	0.88	0.98	0.98	0.89	0.70
GPT2	0.55	0.95	0.91	0.55	0.90	0.96	0.96	0.85	0.92	0.77	0.97	0.95	0.78	0.95	0.98	0.98	0.92	0.96

techniques, classification algorithms, feature selection, and data sampling techniques. To evaluate the models devised for classifying the purpose of the messages in Gitter communications, we have employed evaluation metrics for statistical analysis, box plots for visual representation, and the Friedman test to identify the significant differences among the models.

The Friedman test is a non-parametric test used to determine if there is a significant difference in the average ranks of multiple related samples. In our research, this test was used to test the validity of the following hypothesis:

- **Null Hypothesis** - *The predictive abilities of the models developed using the combination of various word-embedding techniques, classifiers, feature selection, and data sampling techniques are similar.*
- **Alternate Hypothesis** - *The models' predictive abilities developed using various word-embedding techniques, classifiers, feature selection, and data sampling techniques significantly differ.*

B. *Word-Embedding Techniques*

The text messages present in the dataset were converted to a numeric vector representation to develop classification models. The nine-word embedding techniques employed for this purpose are Continuous Bag of Words (CBOW), Skip-Gram (SKG), Global Vectors for Word Representation (GloVe) with 50 dimensions (GLOVE50), 100 dimensions (GLOVE100), and 300 dimensions (GLOVE300), Word2Vec (W2V), fast-Text (FST), Generative Pre-trained Transformer (GPT), and Generative Pre-trained Transformer-2 (GPT2).

Comparison of Word-embedding techniques using Box Plots: Figure 2 provides a graphic representation of the values of the evaluation metrics - Accuracy, Sensitivity, Specificity, and G-Mean of different word embedding techniques applied in the form of Box-plots. While we can observe that the average G-Mean is highest for models based on the GPT-2 word-embedding technique with a value of 0.54, followed by GPT with 0.49, the highest maximum G-Mean is for the models developing using Word2Vec word embeddings with a value of 0.9978, followed by GloVe 300d with a value of 0.9972. The Q3 G-Mean is highest for models with GPT-2, followed by GloVe 300d. Comparing the accuracies of the models provides more stable results as models built using Word2Vec and GloVe 300d emerge as the best-performing models in terms of maximum, Q3, and median accuracy values, while models using GPT and GPT-2 lag behind in this analysis. Since the box plots for the G-Mean values don't paint a clear picture, we rely on the Friedman Rank Test to draw inferences.

Comparison of Word-embedding techniques using the Friedman Test:
The study also employs the Friedman test to evaluate the performance of the models developed using various word embedding techniques. This test aims to test the validity of the null hypothesis, which states that "the different word embedding techniques do not significantly impact the performance of the classification models developed." The test was carried out at a significance level of 0.05 and with nine degrees of freedom. Table III displays the Friedman mean ranks of the G-Mean metric obtained for the various word-embedding algorithms. A lower mean rank indicates a better-performing technique. GloVe 300d has the lowest mean rank of 3.77, followed by Word2Vec, which has a mean rank of 4.05. CBoW has the highest mean rank of 6.2, followed by GPT, which has a mean rank of 6.01. From these observations, it can be inferred that models developed using GloVe and Word2Vec have the highest predictive ability, while the models that use CBoW or GPT suffer the most. Another inference that can be drawn is that models developed with word embeddings with generic pre-trained vectors perform superior for the Gitter-Com dataset compared to those developed with other word-embedding techniques, even those that are domain-specific to software systems.

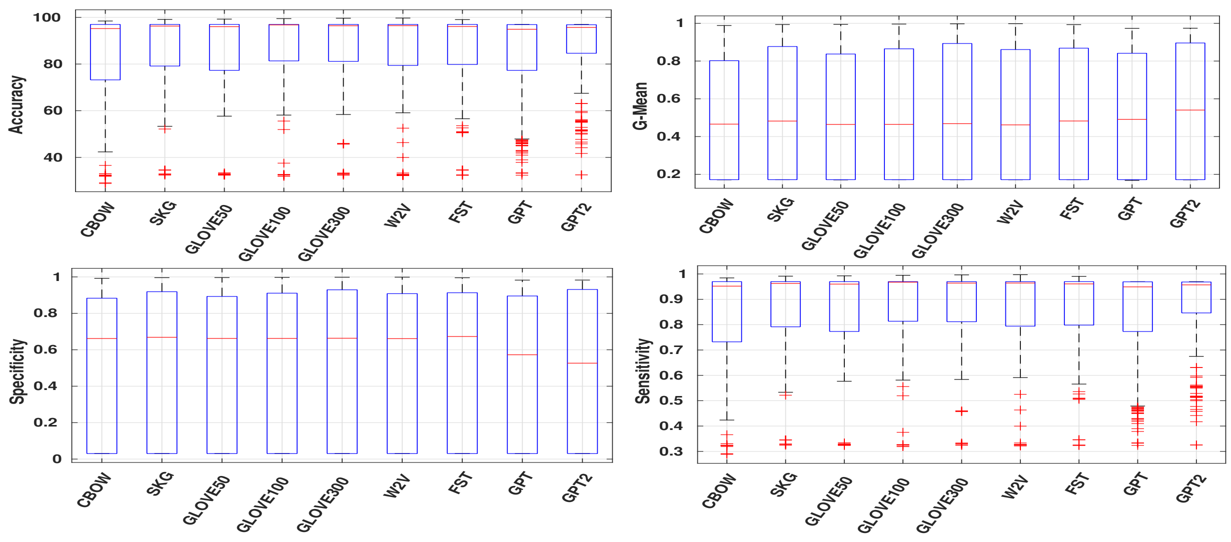

Fig. 2: Performance of Nine Word Embedding

TABLE III: Friedman: Nine Word Embedding

	Accuracy	G-Mean	Specificity	Sensitivity
CBOW	6.11	6.2	6.06	6.11
SKG	4.72	4.25	4.25	4.72
GLOVE50	5.22	5.6	5.63	5.22
GLOVE100	4.57	4.96	4.98	4.57
GLOVE300	3.89	3.77	3.83	3.89
W2V	3.74	4.05	4.12	3.74
FST	5.31	4.71	4.65	5.31
GPT	6.04	6.01	6.01	6.04
GPT2	5.39	5.45	5.47	5.39

C. Feature Selection Techniques

In this work, we have employed two different feature selection techniques - the One-way analysis of variance test and Principal Component Analysis. These techniques were applied to reduce the complexity of the models by retaining or generating the relevant features and discarding the rest. This gave us three different sets of features to compare, the original feature set and the feature seats generated using the feature selection techniques.

Comparison of the Different Sets of Features using Box Plots:
Figure 3 provides a graphic representation of the values of the evaluation metrics - Accuracy, Sensitivity, Specificity, and G-Mean of the models trained using selected sets of features and all features in the form of Box-plots. From the figure, we can draw the inference that the models developed by considering all the features have a marginally better predictive ability for the GitterCom dataset. The average G-Mean of the models developed using the original data is 0.483, with a maximum G-Mean of 0.998 and a Q3 G-Mean of 0.879. The mean G-Mean of the models that use the ANOVA test (0.471) and PCA (0.461) is also quite close to the mean G-Mean of the original data, with ANOVA performing better than PCA for the GitterCom dataset. This is also backed up by the box-plots of accuracies of the models where the models built with all the features slightly out-performs the ANOVA test-based models, which in turn perform much better than models built by considering PCA.

Comparison of the Different Sets of Features using the Friedman Test:
This study also considers the Friedman test to compare the performance of the models using the set of all features or the sets of features generated by the feature selection techniques. This test aims to test the validity of the null hypothesis, which states that "there is no significant difference in the performance of models trained on the data with different sets of features." Table IV displays the Friedman mean ranks of the G-Mean metric obtained for models trained on the original and selected set of features. The models trained using the original feature set have the lowest mean rank of 1.76. The models trained using the feature set generated from the One-way ANOVA test have a slightly higher mean rank of 1.87, and the models trained using PCA have the highest mean rank of 2.37. This analysis indicates that introducing feature selection techniques depreciates the model's performance and that the model works best with all the original features. However, the One-Way ANOVA test models only perform slightly worse than the models with all the features since the mean G-Mean and the mean Friedman ranks are quite close. Hence, if certain word-embedding techniques generate a large number of features for the GitterCom dataset, the One-Way ANOVA test can be considered for feature selection.

TABLE IV: Friedman: Feature Selection Techniques

	Accuracy	G-Mean	Specificity	Sensitivity
AFV	1.86	1.76	1.75	1.86
ANOVA	1.94	1.87	1.88	1.94
PCA	2.2	2.37	2.37	2.2

D. Class Balancing Technique

The Synthetic Minority Oversampling Technique (SMOTE) was used to rectify the class imbalance problem of the dataset, as this technique synthesized data points for the minority classes of the “personal benefits” and “community support” classes.

Comparison of original data and SMOTE synthesized data using Box Plots:
Figure 4 provides a graphic representation of the values of the evaluation metrics - Accuracy, Sensitivity, Specificity, and G-Mean in the form of Box-plots for the models developed using original data and the balanced data obtained from the SMOTE technique. From the figure, we can draw the inference that the models developed by applying the SMOTE technique have a much better predictive ability for the GitterCom dataset compared to the models relying on the original data. The average G-Mean of the models developed using SMOTE is 0.857, with a maximum G-Mean of 0.998 and a Q3 G-Mean of 0.956, which indicates that 25% of the models created using SMOTE have a G-Mean greater than 0.956. Since the

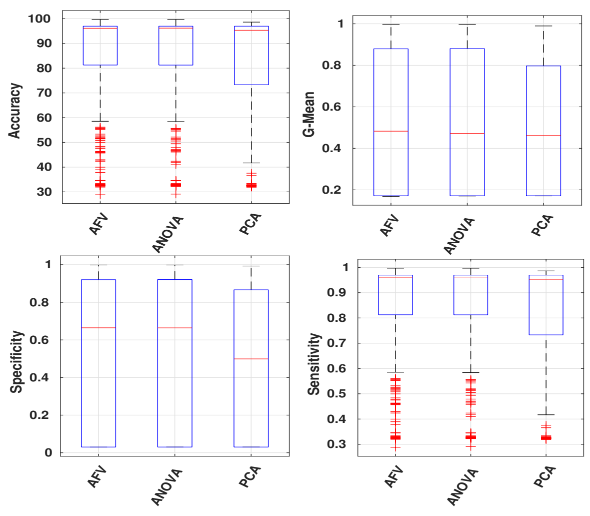

Fig. 3: Performance of Feature Selection Techniques

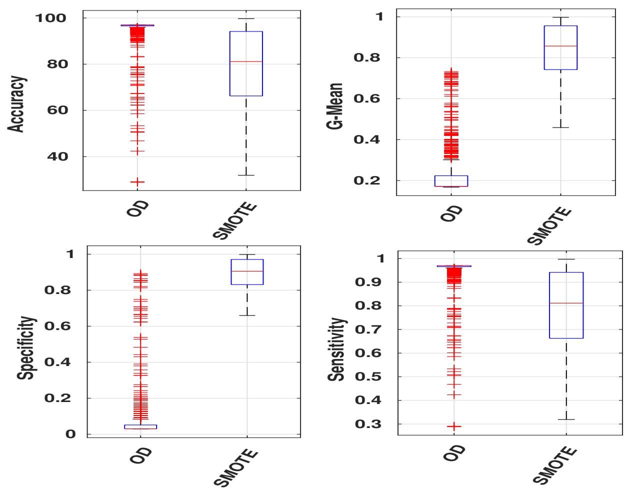

Fig. 4: Performance Class Balancing Technique

original data is highly class-imbalanced, accuracy is not a reliable metric for this analysis since it projects the models with original data to perform better than models with the SMOTE synthesized data.

Comparison of original data and SMOTE synthesized data using the Friedman Test:
This study also employs the Friedman test to compare the performance of the models developed using the original imbalanced dataset and the SMOTE-synthesized dataset. This test aims to test the validity of the null hypothesis, which states that "there is no significant difference in the performance of models trained with dataset having balanced or imbalanced classes, and the algorithm used to balance classes has no significant impact on their performance." Table V displays the Friedman mean ranks of the G-Mean metric obtained for models trained on the original and the SMOTE-balanced dataset. While the latter models have a mean rank of 1, the former models have a mean rank of 2. This is in accordance with the inference obtained from the box plots, thus declaring the use of the SMOTE data sampling technique to improve the performance of the models trained on the GitterCom dataset. The mean ranks of Accuracy provide a different picture where the models trained using the original data have a mean rank of 1.2 which is lower than the mean rank of the models trained using the SMOTE balanced dataset (1.8). This shows the unreliability of the accuracy metric in the presence of class-imbalanced data, thus establishing the G-Mean as the dependable metric in such circumstances.

TABLE V: Friedman: Class Balancing Technique

	Accuracy	**G-Mean**	**Specificity**	**Sensitivity**
OD	1.2	2	1.99	1.2
SMOTE	1.8	1	1.01	1.8

E. Classification Techniques

In this work, we have used seventeen different classification algorithms such as Multinomial Naive Bayes (MNB), Bernoulli's Naive Bayes (BNB), Gaussian Naive Bayes (GNB), Decision Tree (DTC), Logistic Regression (LRC), K-Nearest Neighbours (KNN), KNN with Bagging (KNBG), Multinomial Naive Bayes with Bagging (MNBG), Logistic Regression with Bagging (LRBG), Decision Trees with Bagging (DTBG), Random Forest (RF), Extra Trees (EXTC), Ada Boost (DBG), Gradient Boosting (GRB), Multi-Layer Perceptron with Limited-Memory BFGS (MLPB), SGD (MLPS) and ADAM (MLPA).

Comparison of Classification Techniques using Box Plots:
Figure 5 provides a graphic representation of the values of the evaluation metrics - Accuracy, Sensitivity, Specificity, and G-Mean of different classification algorithms applied in the form of Box-plots. From the figure, we can observe that while tree-based classifiers generally perform better than the rest, the Naive Bayes-based classifiers lag behind in their performance. Random Forest and Extra Trees Classifiers have the highest maximum and Q3 G-Mean values, while Bernoulli and Multinomial Naive Bayes classifiers have the least values. The median G-Mean values paint a slightly different picture as Gaussian Naive Bayes has the highest value, followed by the Decision Tree classifier. The accuracy box plots suggest that Random Forest and Extra Trees classifiers are best performing, followed by Decision Trees with Bagging, while Gaussian Naive Bayes lags behind. Since we can't zero in on the best-performing classifier using the Box Plots, we rely on the Friedman Rank test.

Comparison of Classification techniques using the Friedman Test:
The study also employs the Friedman test to evaluate the performance of the models developed using various classification algorithms. This test aims to test the validity of the null hypothesis, which states that "the choice of the classification algorithms does not significantly impact the performance of the classification models developed." The test was carried out at a significance level of 0.05 and with seventeen degrees of freedom. Table VI displays the Friedman mean ranks of the G-Mean metric obtained for the various classification algorithms. A lower mean rank indicates a better-performing algorithm. We can observe that the Decision Tree classifier has the lowest mean rank of 4.53 among all classifiers, followed by the Extra Trees Classifier at 4.81 and the Random Forest Classifier at 5.46. Bernoulli's Naive Bayes Classifier has the highest rank of 14.13, followed by the Multinomial Naive Bayes classifier with Bagging, with a mean rank of 12.09. Thus, we can infer that the Decision Tree Classifier is the best-performing classification algorithm for the GitterCom dataset, followed by Extra Trees and Random Forest Classifiers, while Bernoulli's Naive Bayes classifier performs the worst for the dataset.

F. One-vs-One and One-vs-rest multi-class classification:

The above-mentioned classification algorithms were implemented using one-vs-one and one-vs-rest multi-class classifi-

TABLE VI: Friedman: Class Balancing Technique

	Accuracy	G-Mean	Specificity	Sensitivity
MNB	10.95	11.3	11.3	10.95
BNB	13.73	14.13	14.13	13.73
GNB	14.52	6.69	6.69	14.52
DTC	10.36	4.53	4.53	10.36
LRC	7.18	10.16	10.16	7.18
KNN	8.6	9.79	9.79	8.6
KNBG	6.23	9.25	9.25	6.23
MNBG	10.45	12.09	12.09	10.45
LRBG	7.87	10.89	10.89	7.87
DTBG	5.85	8.04	8.04	5.85
RF	5.75	5.46	5.46	5.75
EXTC	5.54	4.81	4.81	5.54
ADB	10.34	9.54	9.53	10.34
GRB	11.26	7.45	7.45	11.26
MLPB	10.19	8.95	8.95	10.19
MLPS	6.61	9.63	9.63	6.61
MLPA	7.57	10.31	10.31	7.57

cation strategies, thus generating thirty-four different classifiers for this analysis.

Comparison of the two classification strategies using Box Plots:

Figure 6 provides a graphic representation of the values of the evaluation metrics - Accuracy, Sensitivity, Specificity, and G-Mean of the two classification strategies applied in the form of Box-plots. From these figures, we can observe that the One-vs-Rest slightly out-performs the one-vs-one strategy as the models developed using the former have a maximum G-Mean of 0.9977 and a Q3 value of 0.86 as compared to its counterpart's values of 0.9974 and 0.85. The accuracy box plot also gives similar results as one-vs-rest has slightly higher maximum and median accuracies as compared to models using one-vs-one classification.

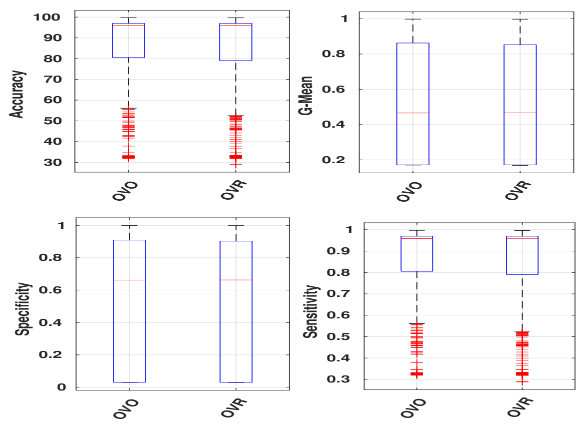

Fig. 6: Performance One-vs-One and One-vs-rest multi-class classification

Comparison of Classification strategies using the Friedman Test:

The study also employs the Friedman test to evaluate the performance of the models developed using the two classification strategies. This test aims to test the validity of the null

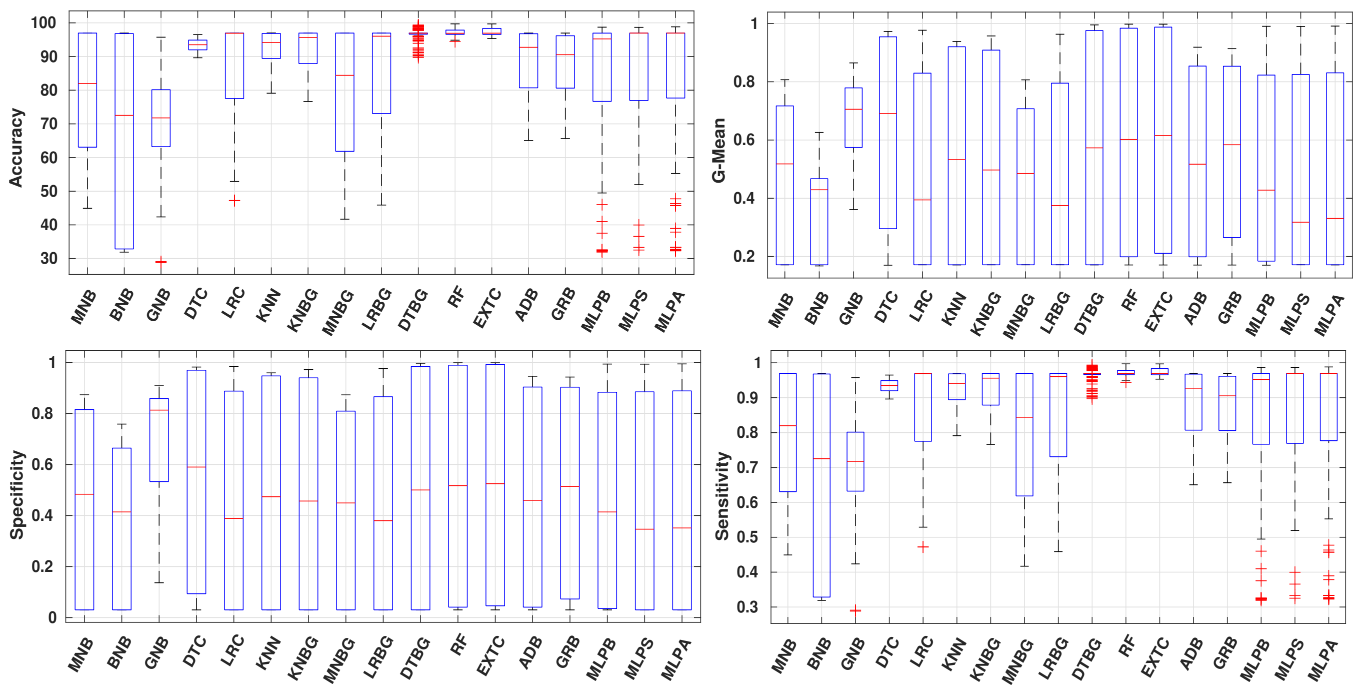

Fig. 5: Performance Classification Techniques

hypothesis, which states that "the choice of the classification strategy does not significantly impact the performance of the classification models developed." The test was carried out at a significance level of 0.05 and with two degrees of freedom. Table VII displays the Friedman mean ranks of the G-Mean metric obtained for the various classification algorithms. We can observe that the models using one-vs-one classification have a slightly lower Friedman mean Rank of 1.47 as compared to the one-vs-rest models' rank of 1.53. This trend is also followed by the mean ranks of accuracies of the two strategies. Thus, the Friedman Rank test concludes that the one-vs-one classification is slightly better than the one-vs-rest classification.

TABLE VII: Friedman: One-vs-One and One-vs-rest

	Accuracy	**G-Mean**	**Specificity**	**Sensitivity**
OVO	1.49	1.47	1.48	1.49
OVR	1.51	1.53	1.52	1.51

VI. Conclusion

Analyzing and classifying the purpose of the messages on software messaging and collaboration platforms such as Gitter provides various benefits, such as analyzing the present open-source development trends and understanding why developers prefer such platforms. Automated message classification can save time and labor and reduce misclassification errors. This paper aims to improve the classification accuracy of the purpose of the messages in the GitterCom dataset by finding the right combination of ML and NLP techniques for the best performance. Various word-embedding techniques, feature selection techniques, classification algorithms, and a data sampling technique were employed in this work. The comparative analysis helps analyze each technique's merits and demerits for analysis of the GitterCom dataset. The key conclusion obtained in this work are:

- Models trained using GloVe 300d and Word2Vec perform superior to models trained with other word-embedding techniques, even the ones that are domain-specific to software systems.
- The application of feature selection techniques slightly degrades the performance of the classifier models. However, the ANOVA test is still a viable alternative in case the computational complexity of the models needs to be improved.
- The class-balancing technique (SMOTE) improved the performance of the models by addressing the class imbalance problem.
- The tree-based classification algorithms outperformed others as the Decision Trees classifier emerged as the best, followed by Random Forest and Extra Trees classifiers.
- The one-vs-one multi-class classification strategy performed better than the one-vs-rest classification for the GitterCom dataset message purpose classification.

While our research focuses on the purpose of the messages, future research could look into insights that can be drawn from messages of each type of purpose. The team-wide purpose messages can be used to analyze collaboration patterns and geographical trends influencing the development projects. Similarly, the personal benefit messages can be used to analyze the commonly faced issues and the extent of help being provided to these users on the platform. Topics of discussion can be analyzed to identify the latest trends and popular technologies in software development. These insights help update the platforms to suit the needs of the developers better, attracting more users and promoting collaboration. Developing such models also helps users understand these trends, which they can incorporate into their own development practices, making them more efficient. Future research to draw insights from the data of such developer communication platforms can be made easier by adapting the techniques and pipelines shown to be more effective and hence recommended by the paper.

VII. Acknowledgements

This research is funded by TestAIng Solutions Pvt. Ltd.

References

[1] E. Parra, A. Ellis, and S. Haiduc, "Gittercom: A dataset of open source developer communications in gitter," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 563–567.

[2] O. Ehsan, S. Hassan, M. E. Mezouar, and Y. Zou, "An empirical study of developer discussions in the gitter platform," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 1, pp. 1–39, 2020.

[3] H. Sahar, A. Hindle, and C.-P. Bezemer, "How are issue reports discussed in gitter chat rooms?" *Journal of Systems and Software*, vol. 172, p. 110852, 2021.

[4] E. Parra, M. Alahmadi, A. Ellis, and S. Haiduc, "A comparative study and analysis of developer communications on slack and gitter," *Empirical Software Engineering*, vol. 27, no. 2, p. 40, 2022.

[5] B. Lin, A. Zagalsky, M.-A. Storey, and A. Serebrenik, "Why developers are slacking off: Understanding how software teams use slack," in *Proceedings of the 19th acm conference on computer supported cooperative work and social computing companion*, 2016, pp. 333–336.

[6] V. Stray, N. B. Moe, and M. Noroozi, "Slack me if you can! using enterprise social networking tools in virtual agile teams," in *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2019, pp. 111–121.

[7] R. Alkadhi, T. Lata, E. Guzmany, and B. Bruegge, "Rationale in development chat messages: an exploratory study," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 436–446.

[8] R. Alkadhi, J. O. Johanssen, E. Guzman, and B. Bruegge, "React: An approach for capturing rationale in chat messages," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2017, pp. 175–180.

[9] S. Beyer, C. Macho, M. Pinzger, and M. Di Penta, "Automatically classifying posts into question categories on stack overflow," in *Proceedings of the 26th Conference on Program Comprehension*, 2018, pp. 211–221.

[10] A. S. M. Venigalla, C. Lakkundi, and S. Chimalakonda, "Sotagger - towards classifying stack overflow posts through contextual tagging (s)," 07 2019, pp. 493–496.

[11] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 11–20.

[12] S. Tiun, U. Mokhtar, S. Bakar, and S. Saad, "Classification of functional and non-functional requirement in software requirement using word2vec and fast text," in *journal of Physics: conference series*, vol. 1529, no. 4. IOP Publishing, 2020, p. 042077.