SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
http://saucis.sakarya.edu.tr/
e-ISSN: 2636-8129
Publisher: Sakarya University
Vol. 7, No. 1, 2024, 46-60
DOI: 10.35377/saucis...1402414

RESEARCH ARTICLE

An Analysis of Intelligent Turkish Text Classification Models for Routing Calls in Call Centers: A Case Study on the Republic of Turkiye Ministry of Trade Call Center

Muammer Özdemir [1], Yasin Ortakcı [2]

[1] Department of Computer Engineering, Karabuk University, Karabük, Türkiye
[2] Department of Computer Engineering, Karabuk University, Karabük, Türkiye

Corresponding author:
Yasin Ortakcı, Karabük Üniversitesi,
Department of Computer Engineering
yasinortakci@karabuk.edu.tr

Article History:
Received: 09.12.2023
Accepted: 14.03.2024
Published Online: 27.04.2024

ABSTRACT
Call centers play a key role in the management of customer relationships in the modern business world. However, the growing demand for their services presents significant challenges, particularly in terms of staffing and handling increasing call volumes. This paper addresses these issues by presenting an AI-driven text classification framework tailored for the Republic of Turkiye Ministry of Trade Call Centre (MTCC), with the aim of automatically routing calls to relevant departments. Using a specific dataset of 20,000 phone call texts collected from the MTCC, the study employs TF-IDF, Word2Vec, and GloVe text vectorization techniques and applies various machine learning algorithms such as K-Nearest Neighbours, Naive Bayes, Support Vector Machines, Adaptive Boosting, Decision Tree and Random Forest for text classification. Through a comprehensive analysis, the study answers key research questions regarding optimal classifiers and vectorization methods. The proposed solution not only improves the efficiency of MTCC's call routing but also provides researchers with practical insights regarding Turkish text classification. The results indicate that a combination of the Random Forest classifier and Word2Vec text vectorization method is the optimal model that can manage to route calls in real-time.

Keywords: Text classification, Word2Vec, GloVe, TF-IDF, Call center

1. Introduction

Call centers play a crucial role in modern business in order to enhance customer relation management. Their goal is to ensure customer satisfaction by providing them with accurate information, responding quickly to inquiries, and dealing effectively with service requests. In recent years, there has been a surge in demand for call center services as a result of technological advances and increases in trading volumes. However, many companies and institutions face challenges in customer relations management due to a lack of qualified staff to handle initial inquiries. This situation leads to delayed and inaccurate problem resolution, resulting in longer wait times and customers having to make repeated calls, even for simple issues. The result is an increase in customer dissatisfaction and a rise in complaints. Furthermore, call center representatives, who are frequently overburdened, are exposed to excessive stress and have to deal with angry customers. Therefore, there is an urgent need for intelligent assistance systems to improve call center representatives' performance and motivation. These factors underline the significance of integrating automation and intelligent support tools into call centers.

In this paper, we develop an AI-driven text classification framework to address the challenges faced in call centers. As a case study, we have focused on the routing of incoming calls to the Republic of Turkiye's Ministry of Trade Call Centre (MTCC) to the relevant departments, taking into account the call text. MTCC receives an average of 10,000 daily calls, which are first answered by call center representatives. These representatives decide whether to handle the call themselves or forward it to the relevant department. However, due to the diverse call content and varying call center representative knowledge and experience levels, they sometimes provide customers with the wrong solutions or direct them to the wrong departments. These errors result in unresolved problems or delayed solutions. These delays not only disrupt the operation of the Ministry of Trade but also lead to economic losses for the country.

To tackle these challenges, this study introduces an innovative solution that utilizes various machine learning techniques to route incoming calls to the relevant departments automatically in the MTCC. The objective is to enhance customer satisfaction by delivering quicker and more precise solutions. To achieve this, we used a dataset of 20,000 sample call texts received by the MTCC and their correctly routed departments. These texts were pre-processed and transformed into numerical representations using TF-IDF (Term Frequency- Inverse Document Frequency), Word2Vec, and GloVe text vectorization techniques. We then applied a variety of machine learning algorithms, including K-Nearest Neighbors (K-NN), Naive Bayes (NB), Support Vector Machines (SVM), Adaptive Boosting (AdaBoost), Decision Tree (DT), and Random Forest (RF) to classify these digitized call transcripts and route the calls to the appropriate departments.

Throughout our experiments, we conducted extensive analysis of text classification algorithms and text vectorization methods. Our primary objective was to find the most effective combination of classifiers and text vectorization methods, which demonstrate superior performance in classifying MTCC call text. Additionally, we investigated the responses to the following research questions (RQ): (RQ1) What is the most appropriate classifier to use in text classification? (RQ2) Which text vectorization technique yields better performance across the classifiers? (RQ3) Which classifier and text vectorization method combination produces the highest text classification results? (RQ4) Which combination of classifiers and text vectorization methods yields optimal results in terms of both runtime efficiency and accuracy? Our observations and findings regarding these matters are elaborated in section 4.4 in detail. Consequently, the key contributions of this study can be summarized as follows:

- We propose an automated system designed to streamline the routing of phone calls within the MTCC, thereby improving the efficiency of call center operations.
- This study presents a comprehensive comparative analysis of text classification performance on Turkish texts using classifiers such as K-NN, NB, SVM, AdaBoost, DT, and RF together with TF-IDF, Word2Vec, and GloVe text vectorization methods.
- Some practical information is provided for customer service representatives to enhance the performance of the proposed intelligent system.

The rest of the paper is organized as follows. Section 2 provides the current literature related to this study. Section 3 presents the methodology of this study, describing the data pre-processing, text vectorization, and classification steps. Section 4 provides comprehensive experiments on the MTCC dataset with its implications. Finally, Section 5 concludes our study and provides the scope for future work.

2. Literature Review

Customer satisfaction (CS) is a crucial objective of marketing research, which focuses on evaluating the satisfaction levels of customers with products and services through their experiences with companies [1-2]. Recent developments in information technology have yielded significant advances in assessing CS. Namely, to ensure CS, much research has focused on enhancing the performance of call center departments through various tools. For instance, Park and Gates conducted research demonstrating the ability to automatically measure CS by analyzing call transcripts. This method allows companies to evaluate satisfaction for each call in almost real-time [3]. Similarly, Chowdhury et al. investigated the predictive capacity of turn-taking as a key factor influencing user satisfaction in verbal conversations [4]. Luque et al. carried out a study aimed at predicting CS through the analysis of various acoustic features extracted from customer service conversations. The study utilized convolutional neural networks to measure satisfaction, with a focus on accentuation-related data in the dialogues. Their proposed method demonstrated superior performance in comparison to traditional systems, in terms of AUC and F-score criteria [5]. Chatterjee et al. categorized problematic and non-problematic phone calls by employing an SVM classifier based on audio features such as meal-frequency cepstral coefficients, energy, voice, and zero-crossing rate. Their system identified the problematic calls with an 87,5% accuracy rate [6].

Customer relation management studies on call centers have mainly concentrated on utilizing machine learning techniques for text classification. In this context, Meinzer et al. used four different machine learning classifiers, namely AdaBoost, K-NN, SVM, and RF, to quantify levels of customer dissatisfaction in the automotive sector. The study found that the SVM method, which utilized the radial basis function as its kernel, demonstrated the greatest accuracy among all classifiers in detecting dissatisfaction, with an accuracy of 88.8% [7]. Liu et al. introduced a model that integrates key utterance analysis and logistic regression algorithms for the classification of service conversations in call centers. The effectiveness of this model was demonstrated in their experiments, particularly with a limited training dataset [8]. Busemann et al. introduced a systematic method to categorize emails in call centers according to their content by leveraging shallow text processing and various machine learning techniques such as lazy learners, SVM, and symbolic eager learners. Their model was seamlessly integrated into an assistance system for call center representatives, improving the overall quality of responses [9]. Galanis et al. developed a technique to extract emotional segments from a call center speech database by incorporating SVM [10]. Emmanuela et al. measured customer satisfaction by applying text classification techniques to user surveys of 549 customers organized in different marketplaces. The study employed K-NN, DT, and NB machine learning methods and found that the DT algorithm had the highest accuracy [11].

The first step in text classification studies is transforming texts into numerical representations [12]. Salminen et al. presented a comprehensive classification analysis for online hate detection using comments from various platforms such as YouTube, Reddit, Wikipedia, and Twitter. They employed a combination of classification algorithms, including LR, NB, SVM, XGBoost, and Artificial Neural Network (ANN), along with diverse feature extraction methods like Bag of Words (BoW), TF-IDF, Word2Vec, and BERT. Their findings revealed that XGBoost emerged as the top-performing model, achieving an F1 Score of 0.92 [13]. Alaoui and Nfaoui developed four different deep learning-based text classification models using CNN and LSTM networks with Word2Vec to detect malicious HTTP web requests. They concluded that LSTM has better performance than the others in terms of classification metrics and training time [14]. In another study, Cahyani and Patasik compared the efficacy of TF-IDF and Word2Vec methods for sentiment analysis of commuter line tweets using SVM and Multinomial NB methods. Their two-step approach first categorized tweets as having or not having emotion, followed by classification into five emotion types: happy, angry, sad, scared, and surprised. The results demonstrated that TF-IDF outperformed Word2Vec across various metrics, enhancing classification performance compared to previous studies [15]. Akuma et al. compared the performance of BoW and TF-IDF feature extraction methods for hate speech detection from live tweets. They combined machine learning methods such as NB, DT, LR, and K-NN with TF-IDF and BoW, evaluating them on the Kaggle Hate Speech and Offensive Language dataset. When the results were evaluated according to precision, recall, f1-score, and accuracy metrics, DT was the most successful machine learning, while TF-IDF outperformed BoW as a feature extraction technique [16].

For example, Öğe and Kayaalp performed sentiment analysis on IMDB comments using BoW, TF-IDF, FastText, and Word2Vec text representation methods. LR and SVM, which produce similar results, achieved 86%, 87%, 87%, and 83% accuracy for BoW, TF-IDF, Word2Vec, and FastText, respectively [17]. Ekici and Takcı integrated Word2Vec and TF-IDF methods with the Gradient Boosting algorithm and compared their performance on a Turkish spam dataset. According to the results of the study, the TF-IDF and Gradient Boosting pair were more successful than the Word2Vec&Gradient Boosting pair and CNN model [18]. Koruyan and Ekeryilmaz employed the TF-IDF approach, alongside LR, SVM, and Stochastic Gradient Descent classifiers, in their study, which categorized the complaints from three prominent consumer electronics retailers in Turkey that were received via the Sikayetvar.com website. The study showed that LR achieved the highest accuracy rate of 80% [19]. Çelik and Koc carried out a study that categorized Turkish news collected from various sources into six distinct groups. The study employed several classifiers such as SVM, NB, LR, RF, and ANN and used TF-IDF, Word2Vec, and FastText word representation methods. The FastText and SVM combination presented the highest accuracy rate of 95.75%, outperforming other methods [20].

Our study distinguishes itself from prior studies through a comprehensive exploration of various text classification techniques, particularly on a Turkish corpus. Additionally, our contribution to the literature includes an in-depth examination of the synergy between these classifiers and diverse text vectorization methods applied to Turkish texts. Furthermore, we introduce an intelligent assistant application within the developed MTCC framework, capable of autonomously directing calls to the pertinent units.

3. Methodology

In this section, we describe our framework, presenting a systematic approach for vectorization and classification of text within call conversation. Our methodology starts with the pre-processing of incoming call text at the MTCC. We applied three different techniques for text digitization: TF-IDF, Word2Vec, and GloVe. After generating text representation, we proceeded with the classification of call text using six distinct machine-learning algorithms: K-NN, NB, SVM, AdaBoost, DT, and RF. We then evaluated the performance of each classifier with respect to the text vectorization methods to identify the most compatible classifier and text vectorization method. Figure 1 depicts the general overview of our framework.

3.1 Data Pre-processing

Our dataset consists of transcripts of telephone conversations between customers and call center representatives at the MTCC. The data pre-processing step of this study encompasses the tasks of data cleaning, tokenization, and lemmatization.

The data cleaning process includes a series of steps. Firstly, it corrects spelling errors and removes empty text, duplicates, and punctuation from the dataset. Moreover, it filters out user-specific data such as identity numbers, phone numbers, and declaration numbers. Additionally, we eliminate stop words, which are typically short, high-frequency words that do not have significant meaning on their own and are often used for sentence structure and cohesion [21-22]. Stop words are generally excluded from Natural Language Processing (NLP) analysis to reduce dimensionality and speed up processing since they have limited semantic contribution to the text [23]. As the MTCC dataset includes Turkish text records, we used the 'Turkish' package of the Natural Language Toolkit (NLTK) library to remove the stop words, such as "ama", "gibi", "fakat",

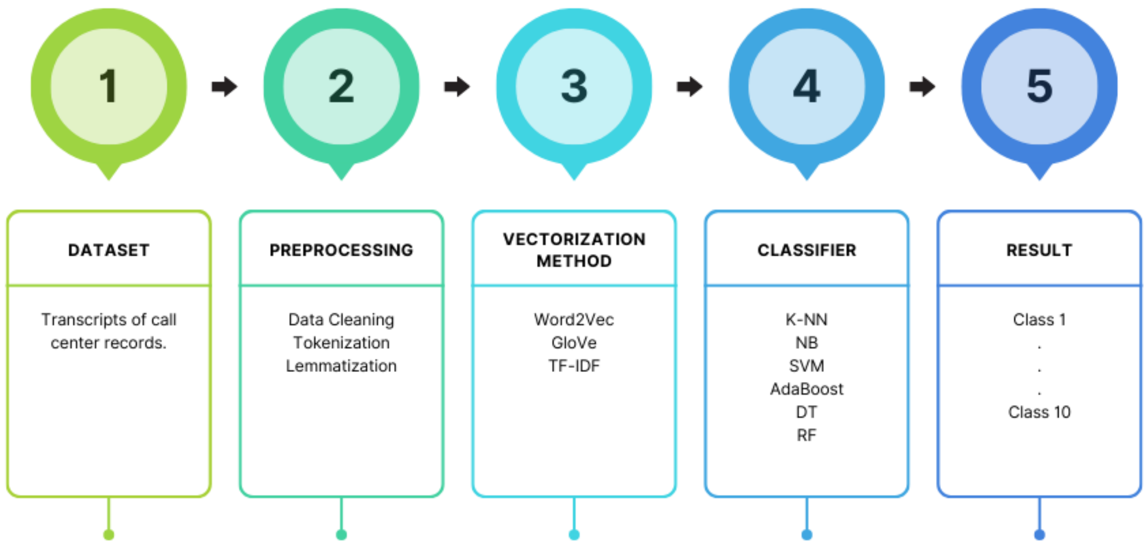


Figure 1 The Implementation Stages of The Proposed Framework for Call Text Classification

"yani", "ya", "mı", "mi", "ne", "ve" and so on. NLTK is an open-source Python library that offers a comprehensive set of NLP functions. Data cleaning ends with the standardization of all text to lowercase.

In the second step of pre-processing, we tokenized the text by segmenting it into words based on spaces. Tokenization is an essential pre-processing technique that breaks text into meaningful units referred to as tokens, such as words, phrases, or symbols [24]. For tokenization, we also utilized the NLTK library.

Finally, we implemented lemmatization to extract the root of the words. Lemmatization is a fundamental NLP technique that modifies or eliminates suffixes on a word to obtain its basic form, known as the 'lemma' [25]. In this study, we used the 'tr' class from Simplemma, an open-source Python library designed to identify the root or basic form of words [26].

3.2 Text Vectorization

After completing the data pre-processing, we applied three distinct methods for converting text to vector representations: TF-IDF, Word2Vec, and GloVe. These methods transform the textual data into numerical representations, allowing us to address NLP problems through mathematical approaches.

3.2.1 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF is one of the most widely used text representation methods, providing a statistical metric to quantify the mathematical significance of words within a given corpus. TF-IDF considers both the frequency of a word within a specific document and how rarely it appears across all documents. The TF-IDF score for a word in a document is calculated by multiplying its Term Frequency (TF) and the Inverse Document Frequency (IDF) scores, resulting in a score that ranges from 0 to 1.

To illustrate the computation of a basic TF-IDF value in a given corpus, let's consider an example. TF refers to the division of the frequency of a given word (t) in a particular document (*d*), divided by the total number of words in that document (*Z*), as shown in Eq. (1). On the other hand, the IDF involves the calculation of the logarithm of the total number of documents (*N*) in the corpus (*C*), divided by the number of documents in which the word (*t*) appears, as shown in Eq. (2). The TF-IDF value is then obtained by multiplying these two values, as in Eq. (3). Consequently, words that frequently appear within a specific document but rarely across all documents receive a high score, indicating their significance in the text representation. Conversely, words that frequently appear across all documents receive a low score, showing their limited influence [27].

$$TF(t,d) = count(t,d) \, / \, Z \tag{1}$$

$$IDF\,(t,D) = log\left(\frac{N}{count\,(\,d\,\in\,D\,:\,t\,\in\,d\,)}\right) \tag{2}$$

$$TF\ - IDF\,(t,d,D) = TF(t,d) * IDF(t,D) \tag{3}$$

where:

$count(t,d)$: *Number of a given word (t) in a specific document.*

Z: Total number of words in a particular document

D: The corpus including all documents

N: Total number of documents in the corpus

$count(d \epsilon D: t \epsilon d)$: *Number of documents where the t term appears.*

For instance, if the term "kapikule" appears four times in a document that is composed of a total of 50 words, then the TF value would be 0.08. Additionally, if "kapikule" is found in 10 out of 500 documents in the corpus, then the IDF value would be calculated as 1.69. By multiplying both values, the resulting TF-IDF value equals 0.135.

3.2.2 Word2Vec

Word2Vec, developed by Mikolov et al. in 2013, is a text representation technique designed to improve the effectiveness of the Google search engine [28]. Its primary objective is to generate word representations using ANNs in a prediction-based manner. Unlike TF-IDF, which encodes text as high-dimensional sparse matrices, Word2Vec produces dense vector word embeddings for each word in the text. It also places similar words close together in the vector space. Word embeddings offer two key advantages: a reduction in computational cost due to lower dimensionality and improved performance in NLP tasks by effectively grouping words that are semantically similar [29-30]. Word2Vec generates word embeddings by capturing the semantic meaning of words based on their context within the text. To achieve this, it employs two different neural network models: Continuous Bag-of-Words (CBOW) and Skip-gram [31].

The CBOW model aims to forecast the central word by considering the surrounding contextual words, which usually include a few words preceding and following the targeted word in the text. CBOW employs a defined window size that specifies the number of neighboring words taken into account. The words within this window collectively contribute to the prediction of the corresponding word. Figure 2 visually represents the CBOW architecture, with w(t) denoting the current word and terms ranging from w(t-2) to w(t+2) representing the surrounding words within the window size of the corresponding word, w(t). In this neural network, the hidden layer is a typical, fully connected, dense layer, whereas the output layer calculates the probabilities for the target word in the vocabulary.

The skip-gram model takes a word in a text as the center of the window size, then predicts the neighboring words in this window as outputs. The central word in the window is denoted as w(t), while the surrounding words are represented with a range from w(t-2) to w(t+2) as shown in Figure 3. The skip-gram model predicts the neighboring words around a single word, considering the context. This characteristic makes the skip-gram model advantageous in representing less common or

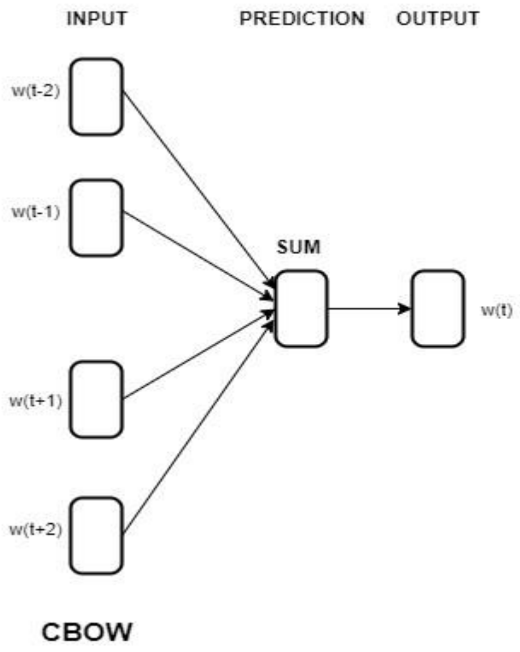


Figure 2 The General Architecture of CBOW

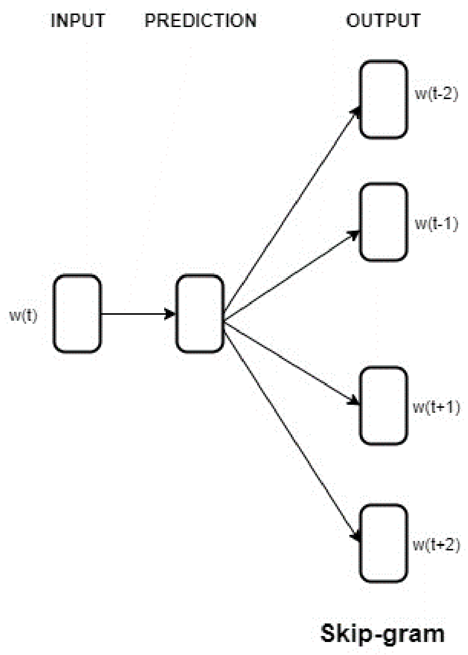


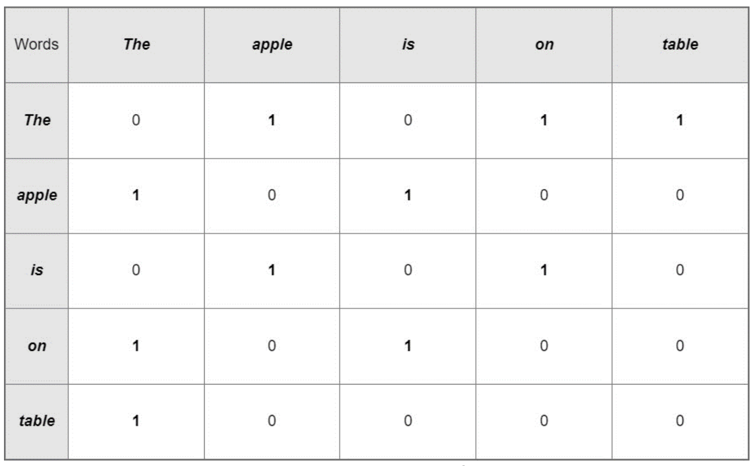
Figure 3 The General Architecture of Skip-Gram

terms, enhancing its efficiency compared to CBOW. Skip-gram is also more effective than CBOW in handling large datasets. These reasons and the fact that Skip-Gram produces better results in a shorter time than CBOW in our preliminary test led us to use Skip-Gram in this study. In our preliminary test, the runtimes for Skip-gram were 4.64, 0.65, 1.6, 4.61, 5.29, and 2.54 seconds respectively for K-NN, SVM, NB, AdaBoost, DT, and RF, whereas for CBOW, the corresponding times were 4.86, 0.66, 2.1, 4.70, 13.03, and 6.66 seconds. These results clearly show that Skip-gram exhibits shorter execution times across all classifiers compared to CBOW.

3.2.3 GloVe

GloVe, an acronym for "Global Vectors," refers to a word embedding model that was introduced by Pennington et al. [32]. In GloVe, the co-occurrence matrix (X) is created from a corpus containing v words and presented in a $v \times v$ matrix format. Each element in this matrix (X_{ij}) denotes the frequency of co-occurrence between $word\ i$ and $word\ j$. For instance, the co-occurrence matrix for the given sentence "The apple is on the table" with a window size of one is given in Figure 4.

This model distinguishes itself from Word2Vec by presenting a novel unbiased objective function that employs probability statistics for word prediction. GloVe generates precise word vectors by minimizing the prediction errors through an error function, which combines both local statistics derived from the corresponding sentence and global statistics obtained from the entire corpus.



Words	***The***	***apple***	***is***	***on***	***table***
The	0	**1**	0	**1**	**1**
apple	**1**	0	**1**	0	0
is	0	**1**	0	**1**	0
on	**1**	0	**1**	0	0
table	**1**	0	0	0	0

Figure 4 Glove Co-Occurrence Matrix of Given Sample Sentence

3.3 Text Classification

This stage focuses on routing the call center records, the numerical representations of which have been generated in the preceding phase, to the respective departments. Within this framework, six different machine learning algorithms, K-NN, NB, SVM, AdaBoost, DT, and RF, are trained using the MTCC dataset to establish text classification models. This study presents a comprehensive analysis of the effectiveness of each classification model, taking into account its harmony with TF-IDF, Word2Vec, and GloVe methods.

3.3.1 K-Nearest Neighbors (K-NN)

The k-NN algorithm, first formulated by Evelyn Fix and Joseph Hodges in 1951 and later extended by Thomas Cover, is a non-parametric supervised learning approach [33]. This algorithm uses a sample dataset with known classes to determine the class membership of a new sample. The calculation is performed by considering the distances between the new data point and the existing dataset, ultimately determining the final class of the point through the application of the k nearest neighbor majority approach. Specifically, the K-NN classifier uses the Euclidean distance metric to quantify the proximity between data points.

3.3.2 Naïve Bayes (NB)

NB, named after the mathematician Thomas Bayes, classifies data using the principles of probability [34]. NB calculates the probability of each class separately for a given sample and assigns the sample to the class with the highest probability. This technique is often used in various scenarios based on text classification, such as spam filtering and sentiment analysis. In general, versions of Multinomial NB, Bernoulli NB, and Gaussian NB are commonly used NB types.

3.3.3 Support Vector Machines (SVM)

SVM, proposed by Cortes and Vapnik (1995), has been extensively employed by researchers to address classification problems. SVM is a supervised machine learning algorithm that is used for classification and regression tasks [35]. This algorithm projects the data onto a high-dimensional feature space using kernels and identifies hyperplanes that divide the data into separate categories [36]. When identifying these hyperplanes, the aim is to maximize the distance between the closest points within different categories. This approach provides a robust classification model that is more resistant to noise and outliers.

3.3.4 Decision Tree (DT)

DT is a hierarchical tree-like structure based on the features of the input data. Each node in the tree represents a particular feature, while the edges leading from the node represent grouping components [37]. DTs can capture non-linear relationships between the input features and the target class. However, they can suffer from overfitting in cases where the tree is too complex or the training data is too noisy, leading to poor classification performance. To overcome these problems, pruning or ensembling methods can be applied.

3.3.5 Adaptive Boosting (AdaBoost)

AdaBoost is a popular ensemble machine-learning algorithm used for classification problems. It creates a robust classification model by combining multiple weak classifiers. AdaBoost is known for its ability to solve complex classification problems and handle noisy and imbalanced data. It has been successfully applied in various fields, such as object detection, face recognition, and medical diagnosis [38].

3.3.6 Random Forest (RF)

RF is an ensemble learning algorithm consisting of multiple DTs, each trained on a different randomized subset of the dataset [39]. RF runs the standard DT algorithm on each subset and makes its final prediction by combining the results of the DT sets. In this combining process, RF applies the majority voting or the weighted voting methods to the class labels obtained from all trees. RF is a popular and powerful algorithm and has been successfully applied to many classification problems, such as image and speech recognition.

4. Experimental Results

This section presents a set of experiments and their results to evaluate the effectiveness of the proposed text classification framework. To ensure the reliability of the results, each classifier, in combination with three text vectorization methods, was run 10 times, and the averages were calculated along with their respective standard deviations.

To increase the robustness of the classifier models, we also incorporated k-fold cross-validation, a statistical technique essential for impartially evaluating the performance of machine learning algorithms. Rather than assessing a model solely on a single training/test subset, cross-validation employs k-fold validation, dividing the dataset into k different training/test subsets. This approach effectively mitigates the overfitting issues, a common concern in machine learning, and produces more reliable results. By training the model on varying training/test partitions of the same dataset, cross-validation ensures a more uniform and robust training process. In this study, we applied 5-fold cross-validation, meaning that the model creation was repeated five times. Within each fold, the dataset was divided into training and test subsets, with a split of 80% to 20%.

4.1 Setup

The experiments were carried out on a PC with the following specifications: an Intel (R) Core (TM) i7-10700 CPU @ 2.90GHz, with 8 cores and 16 logical processors, supported by 32 GB of RAM and a 500 GB SSD. The proposed framework was developed using the Python programming language within the Spyder environment (Anaconda3 version 4.10.3). We leveraged the sci-kit-learn library version 0.24.2[1] to implement the classification algorithms and cross-validation technique and to calculate evaluation metrics. The optimal parameters of the classifiers used in the experiments are listed in Table 1.

Table 1 Optimal Parameter Values of Each Classifier Used in The Experiments

Method	Parameter	Value
K-NN	n-neighbors	5
NB	alpha	1.0
SVM	kernel	poly
	degree	3
	c	1.75
	gamma	scale
	coef0	0.0
AdaBoost	base_estimator	ExtraTreeClassifier
	n_estimators	50
	learning_rate	1.0
DT	min_samples_split	default=2
	min_samples_leaf	default=1
RF	min_samples_split	default=2
	min_samples_leaf	default=1
	n_estimators	15

4.2 Dataset

Our dataset consists of 20,000 records, including transcripts of phone calls to the MTCC. These records are obtained from actual calls made to the call center and are converted into text and forwarded to the relevant department by customer representatives. The department to which the call is directed becomes the label of the records in the dataset. The records and labels used in this study were validated by experts from the Ministry of Commerce to ensure their accuracy and suitability as a dataset. However, these records are in their raw and unedited form, with no imposed character limit. This comprehensive collection is sourced equally from 10 different departments within the Ministry of Trade. The distribution of these records across the departments and the average number of words and characters in the records of each department are shown in Table 2. Our dataset consists of 20,000 records, including transcripts of phone calls to the MTCC.

Table 2 The Distribution of Call Records Across the Department

Department Label	Department Name	Number of Calls	Average Word Count	Average Character Count
0	Center of Ministry	2000	31	231
1	Retail Trade	2000	29	226
2	MERSİS	2000	37	281
3	Subscription Agreements	2000	46	329
4	Unfair Commercial Practices	2000	31	239
5	After Sales Services	2000	44	320
6	Department Of Exemptions	2000	34	272
7	Consumer Arbitration Committee App Processes	2000	43	318
8	Distance Sales	2000	44	325
9	Defective Goods and Services	2000	33	248

[1] https://scikit-learn.org/stable/user_guide.html

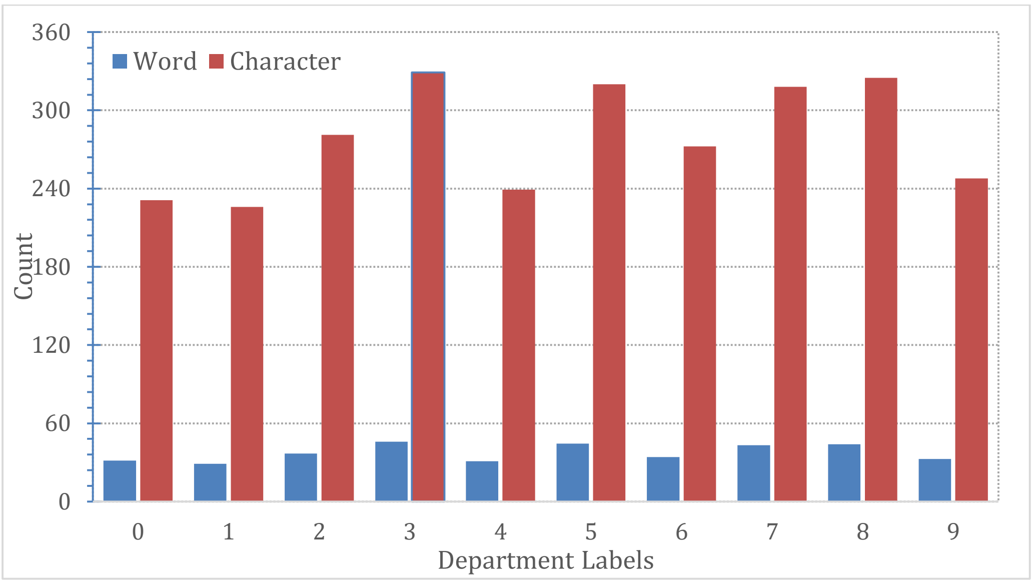


Figure 5 Distribution of the Average Count of Words and Characters in the Call Text across the 10 Departments

Figure 5 illustrates the average number of words and characters present in the texts of phone calls across the 10 departments outlined in Table 2. When we examine the average word and character counts across all departments, it is evident that there is no significant difference that might negatively affect the classification of call texts. This reasonable distribution within our newly created dataset contributes to improving the prediction accuracy of text classification and building a robust model.

4.3 Performance Metrics

To evaluate the effectiveness of text classification algorithms in our experiments, we employed accuracy, precision, recall, and f1-score metrics, which are derived from the confusion matrix. The confusion matrix is a versatile tool, applicable to both binary and multi-class classification problems, that provides a quantitative representation of predicted values compared to their actual values [40]. The confusion matrix enumerates True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) instances, encapsulating the sample counts associated with the results of a classification process. TP represents the number of correctly classified positive examples, while TN denotes the number of correctly classified negative examples. FP signifies the number of examples that are classified as positive but are negative, and FN indicates the number of examples that are classified as negative but are positive.

The accuracy metric, as outlined by Eq. (4), determines the proportion of accurate predictions made by the model out of the total number of predictions. Precision signifies the positive predictive value, which measures the ratio of true positive samples to the total number of positives predicted by the model as in Eq. (5). Recall represents the degree to which positive examples are accurately predicted (Eq. 6). The f1-score is the harmonic mean of recall and precision (Eq. 7).

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \tag{4}$$

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

$$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{7}$$

4.4 Results and Discussion

In the experiments, we ran a total of 18 text classification models, each a combination of a classifier algorithm and a text vectorization method, 10 times each on the MTCC call text. Figure 6 displays the average accuracy values of all models with their standard deviations (SDs) on the bar chart. The small size of the SD bars for all models underlines the consistent results of these models. In addition, we conducted a two-way ANOVA statistical test of 6 (classifiers: AdaBoost, DT, KNN, NB, RF, and SVM) x 3 (vectorizers: Glove, TF-IDF, and Word2Vec) to analyze the variance between the models. We only considered the accuracy as other performance metrics have parallel results with accuracy. The results of the variance analysis were as follows:

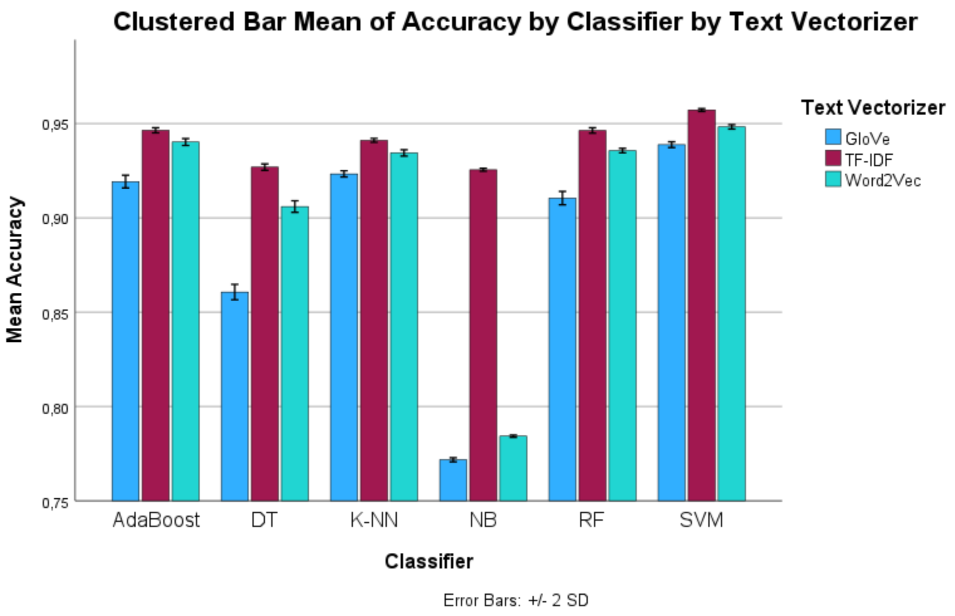


Figure 6 Mean Accuracy Values with Standart Deviation Bars for Each Model

- The results for the classifier showed a significant main effect with **F(5,162)=58209.99, p<0.001, partial η^2 =0.999**, indicating that the accuracy varies between different classifiers.
- The main effect of the vectorizer method was also significant with **F(2,162)=41684.37, p<0.001, partial η^2=0.998**. Namely, each vectorizer has significantly different accuracy results from each other with **Mean$_{GloVe}$=0.887, SD$_{GloVe}$=0.057; Mean$_{TF\text{-}IDF}$=0.940, SD $_{TF\text{-}IDF}$=0.011; and Mean$_{Word2Vec}$=0.908, SD $_{TF\text{-}IDF}$ =0.057** values.
- The interaction effect for classifier × vectorizer was also significant with **F(10,162)=9342.70, p<0.001, partial η^2=0.998**, meaning that the effect of a classifier on accuracy depended on the text vectorizer method.

After confirming that there were significant differences between the models, we performed comparative analyses of these models. When comparing the accuracy values of the classifiers for TF-IDF, the models perform in the following order: SVM, AdaBoost, RF, K-NN, DT, and NB. This ranking remains the same for Word2Vec. In the case of GloVe, the order changes to SVM, K-NN, AdaBoost, RF, DT, and NB. In addressing RQ1, it can be concluded that SVM consistently achieved the highest accuracy, while NB consistently yielded the lowest accuracy regardless of the text vectorization method. Furthermore, it is evident in Figure 6 that SVM, K-NN, AdaBoost, and RF exhibit comparable classification performance, with SVM consistently demonstrating superior performance.

In addition to the accuracy, Table 3 elaborates on the classification results with the metrics precision, recall, f1-score, as well as the runtime of each model obtained when testing a sample text set. The values presented in Table 3 are the averages derived from 10 distinct experimental runs. Regarding the efficiency of the text vectorization methods, TF-IDF consistently produced the highest metrics across all classifiers, followed by Word2Vec and GloVe. The performance order of the text vectorization methods in all classifiers was consistently TF-IDF, Word2Vec, and GloVe as shown in Figure 7. This finding provides a reliable insight into the selection of text vectorization methods as well as answers to our RQ2.

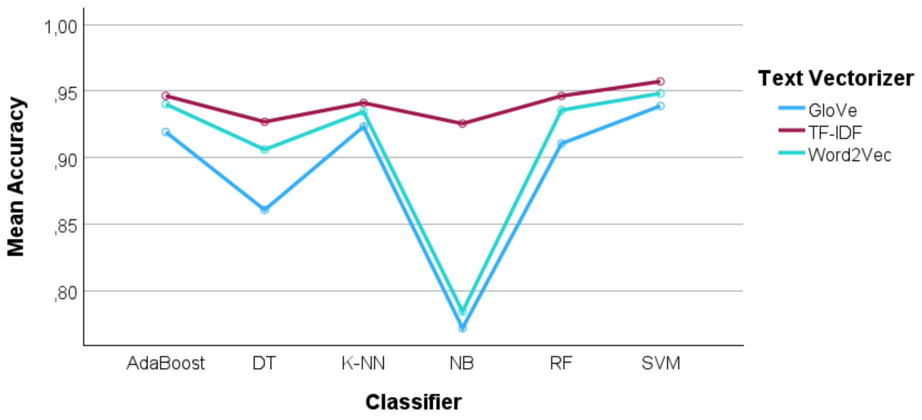


Figure 7 Mean Accuracy Value of Classifiers across the Text Vectorizers

To find the answer to RQ3, we compared the experimental results of 18 different classifier and text vectorizer combinations. As a result, the SVM&TF-IDF combination showed superior performance, achieving 95.7% accuracy, 95.8% precision, 95.6% recall, and 95.7% f1-score in accurately classifying the call text into the corresponding departments. Conversely, the NB classifier showed the lowest performance of all three text vectorization methods for all classification metrics. In particular, the combination of NB with the Word2Vec and GloVe methods fell significantly below the average performance observed with the other methods. On the other hand, the classification performance of the models within our framework closely aligns, except for the combinations of NB&Word2Vec, NB&GloVe, and DT&GloVe.

When comparing runtimes, it is evident that the Word2Vec method has the shortest runtime for all classifiers except K-NN. The DT&Word2Vec combination has the shortest runtime (0.095 sec) among the models, but its accuracy (0.906) is approximately 5% lower than the maximum accuracy observed in the experiments. On the other hand, K-NN and AdaBoost were the least efficient classifiers in terms of runtime. The combination of SVM&TF-IDF achieved the highest accuracy rate of 0.957 with a reasonable runtime averaging 0.937 seconds. The SVM&Word2Vec pair reached an accuracy rate of only about 1% less than SVM&TF-IDF, with an average accuracy of 0.948. Notably, its runtime averages 0.373 seconds, nearly three times less than SVM &TF-IDF. Therefore, SVM&Word2Vec also emerges as one of the most optimal models in terms of both runtime efficiency and accuracy. Regarding RQ4, the experimental results indicate that combinations such as SVM&Word2Vec or SVM&TF-IDF demonstrate the highest levels of accuracy and the most efficient runtimes across all models.

Table 3 Mean Performance Metric Results for 10 Different Runs of Each Model and the Mean Runtimes of Each Model at the Test on a Sample Text Set.

Classifier	Vectorizer	Accuracy ±SD	Precision	Recall	F1-Score	Runtime (sec)
AdaBoost	TF-IDF	0,946 ±0,0007	0,950	0,947	0,948	5.792
	Word2Vec	0,940 ±0,0009	0,944	0,940	0,941	1.321
	GloVe	0,919 ±0,0017	0,927	0,917	0,920	1.201
DT	TF-IDF	0,927 ±0,0009	0,927	0,927	0,927	0.294
	Word2Vec	0,906 ±0,0015	0,906	0,906	0,906	0.095
	GloVe	0,861 ±0,0020	0,860	0,862	0,860	0.119
K-NN	TF-IDF	0,941 ±0,0005	0,940	0,941	0,940	1.863
	Word2Vec	0,934 ±0,0008	0,935	0,934	0,934	1.884
	GloVe	0,923 ±0,0008	0,923	0,923	0,921	1.959
NB	TF-IDF	0,926 ±0,0004	0,925	0,926	0,925	0.128
	Word2Vec	0,784 ±0,0003	0,796	0,784	0,788	0.109
	GloVe	0,772 ±0,0005	0,786	0,771	0,775	0.131
RF	TF-IDF	0,946 ±0,0007	0,948	0,946	0,947	1.122
	Word2Vec	0,936 ±0,0006	0,938	0,936	0,937	0.706
	GloVe	0,910 ±0,0018	0,918	0,910	0,912	0.724
SVM	TF-IDF	**0,957** ±0,0004	**0,958**	**0,957**	**0,957**	0.937
	Word2Vec	0,948 ±0,0006	0,949	0,948	0,948	0.373
	GloVe	0,939 ±0,0008	0,939	0,939	0,939	0.443

Figures 8, 9, and 10 present the confusion matrices for each classifier, obtained by using the TF-IDF, Word2Vec, and GloVe methods, respectively. Within these matrices, numerical labels ranging from 0 to 9 are assigned to the departments outlined in Table 2. The rows indicate the actual department numbers, while the columns represent the predicted department numbers. These matrices provide a comprehensive overview of the classification models' outcomes, considering the correct and incorrect distribution of calls across the departments. For instance, in Figure 7. a, corresponding to the SVM&TF-IDF combination that achieved the highest accuracy in the experiments, 1793 out of 2000 calls for Department 0 (Centre of Ministry) are correctly classified. In contrast, the other 207 calls were misclassified. Specifically, 201 of these calls were assigned to Department 1 (Retail Trade), while 6 calls were assigned to Department 5 (After Sales Services), as incorrect predictions.

Furthermore, these confusion matrices provide some practical information for customer service representatives to improve the performance of the proposed intelligent system. Namely, upon examining all the confusion matrices in Figures 8, 9, and 10, a consistent pattern emerges; each model has the highest error rates in predicting Department 1 (Retail Trade). This finding suggests that call center representatives should use more precise terminology when referring to the Retail Trade Department. Thus, the models' error rates are likely to decrease, leading to more effective and accurate classification results.

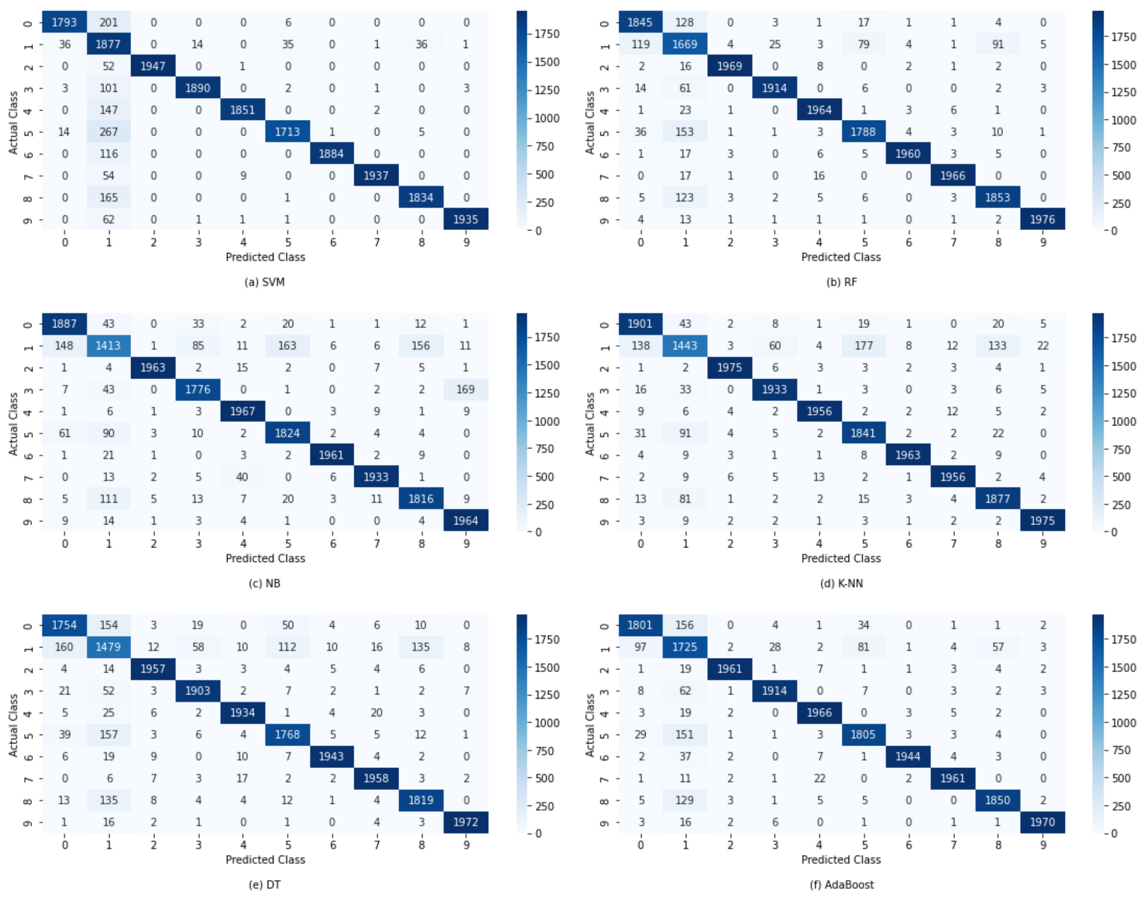


Figure 8 TF-IDF Confusion Matrices for All Classifiers

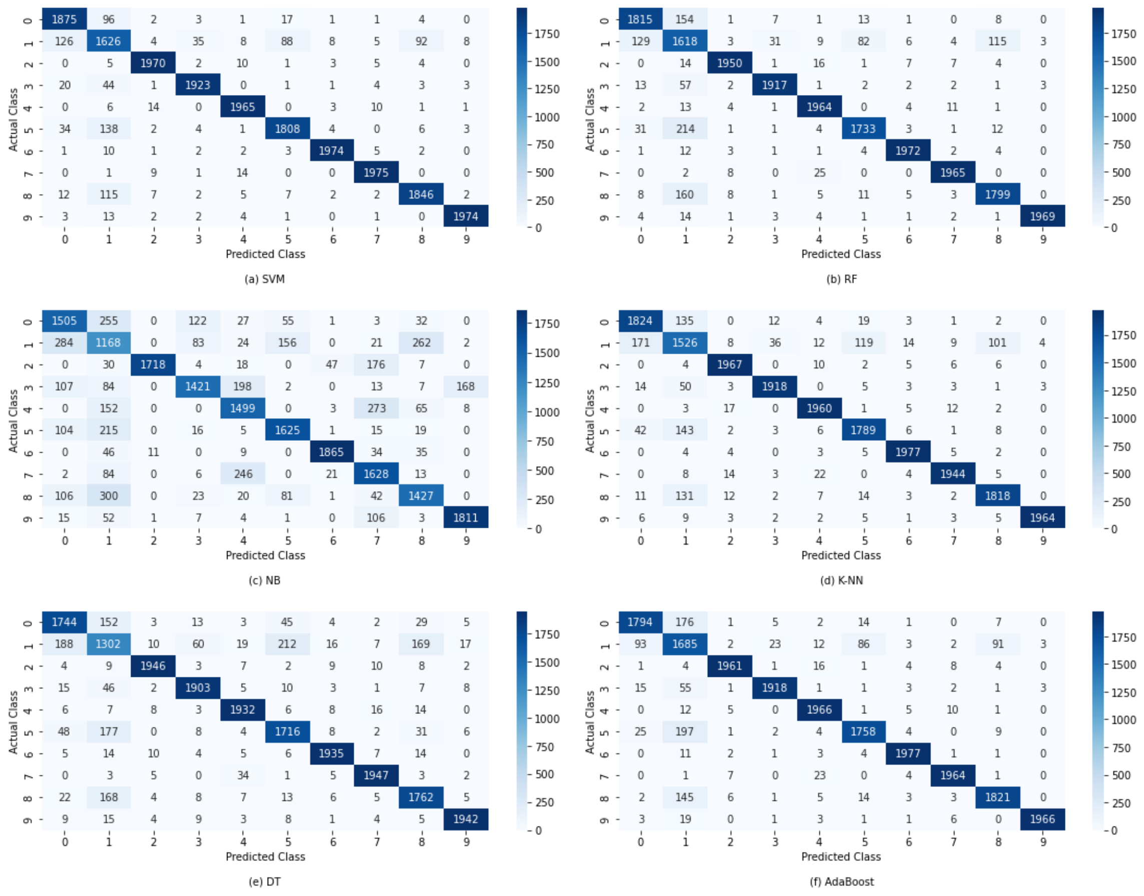


Figure 9 Word2Vec Confusion Matrices for All Classifiers

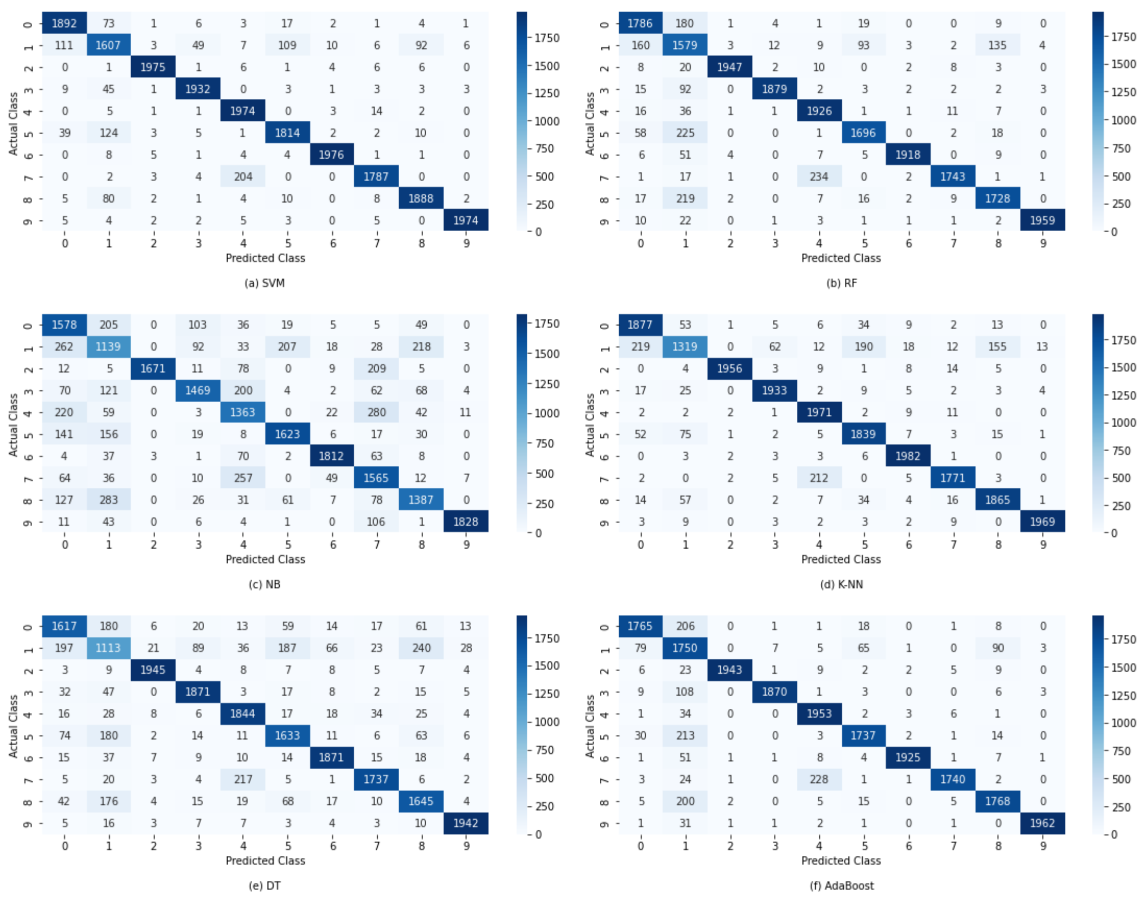


Figure 10 Glove Confusion Matrices for All Classifiers

4. Conclusion

The manual routing of incoming calls in call centers often results in errors, posing challenges to effective customer relationship management. To address these issues, this study introduces an intelligent call center framework. This framework takes the data of MTCC, which receives an average of 10,000 daily calls, as a case study and transforms the transcripts of the incoming calls into numerical representations with TF-IDF, Word2Vec, and GloVe text vectorization methods. Subsequently, six different machine learning algorithms, such as K-NN, NB, SVM, AdaBoost, DT, and RF, are used to classify the call text records to be directed to the relevant department.

In this study, we conducted a series of comprehensive experiments to identify the most successful and practical combination of classifier and text vectorization methods. The results provide valuable insights into classification algorithms and text vectorization techniques applicable to call centers. While SVM emerges as the most successful classifier, TF-IDF outperforms the other text vectorization methods. Considering both runtime and classification performance, the combination of SVM&TF-IDF and SVM&Word2Vec emerge as the most suitable models to serve as an intelligent assistant in MTCC. As a result, the proposed system was able to provide a real-time, automated, Turkish language-oriented solution for call centers.

The complexity of Turkish, in which suffixes can change the meanings of words, poses a significant constraint on word embedding techniques such as Word2Vec and GloVe. Further research is required to address this limitation in processing Turkish text. On the other hand, as a future plan, large language models, which have become very popular recently, can be used as text embedding methods to investigate their success in the classification of Turkish texts.

Acknowledgments

We would like to thank the General Directorate of Information Technologies of the Ministry of Trade in Turkiye for generously providing access to the call center data for the purposes of this study.

References

[1] P. G. Patterson, L. W. Johnson and R. A. Spreng, “Modeling the Determinants of Customer Satisfaction for Business-to-Business Professional Services,” *J. Acad. Mark. Sci.*, vol. 25, no. 1, pp. 4–17, 1996, doi: 10.1177/0092070397251002.

[2] V. Pallotta, R. Delmonte, L. Vrieling and D. Walker, “Interaction Mining: The new frontier of Call Center Analytics,” in Proc. CEUR Workshop in DART@ AI* IA., vol. 771, Sep. 2011, pp. 1-12.

[3] Y. Park and S. C. Gates, “Towards real-time measurement of customer satisfaction using automatically generated call transcripts,” in *Proc. Int. Conf. Inf. Knowl. Manag.*, vol. 24754, 2009, pp. 1387–1396, doi: 10.1145/1645953.1646128.

[4] S. A. Chowdhury, E. A. Stepanov and G. Riccardi, “Predicting user satisfaction from turn-taking in spoken conversations,” presented at the INTERSPEECH, San Francisco, USA, Sept. 8-12, 2016, pp. 2910–2914, doi: 10.21437/Interspeech.2016-859.

[5] J. Luque, C. Segura, A. Sanchez, M. Umbert and L. A. Galindo, “The role of linguistic and prosodic cues on the prediction of self-reported satisfaction in contact centre phone calls,” presented at the INTERSPEECH, Stockholm, Sweden, Aug. 20-24, 2017, pp. 2346–2350, doi: 10.21437/Interspeech.2017-424.

[6] J. Chatterjee, A. Saxena and G. Vyas, “An automatic and robust system for identification of problematic call centre conversations,” presented at the *Int. Conf. Micro-Electronics Telecommun. Eng. (ICMETE)*, Ghaziabad, India, Sept. 22-23, 2016, pp. 325–330, doi: 10.1109/ICMETE.2016.48.

[7] S. Meinzer, U. Jensen, A. Thamm, J. Hornegger and B. M. Eskofier, “Can machine learning techniques predict customer dissatisfaction? A feasibility study for the automotive industry,” *Artif. Intell. Res.*, vol. 6, no. 1, p. 80-90, Dec. 2016, doi: 10.5430/air.v6n1p80.

[8] Y. Liu, B. Cao, K. Ma, and J. Fan, “Improving the classification of call center service dialogue with key utterences,” *Wirel. Networks*, vol. 27, no. 5, pp. 3395–3406, 2021, doi: 10.1007/s11276-021-02573-7.

[9] S. Busemann, S. Schmeier, and R. G. Arens, “Message classification in the call center”, in *Proc. Sixth Applied Natural Language Processing Conference*, 2000, pp. 158–165, doi: 10.3115/974147.974169.

[10] D. Galanis, S. Karabetsos, M. Koutsombogera, H. Papageorgiou, A. Esposito and M. T. Riviello, “Classification of emotional speech units in call centre interactions,” in *Proc. 4th IEEE Int. Conf. Cogn. Infocommunications (CogInfoCom)*. Proc., 2013, pp. 403–406, doi: 10.1109/CogInfoCom.2013.6719279.

[11] E. P. Emmanuela, F. K. Tjendra, S. Kezia and D. Suryani, "Classification of Customer Satisfaction in Marketplace," presented at the *2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE), Jakarta,* ID, Feb. 16, 2023, doi: 10.1109/ICCoSITE57641.2023.10127788.

[12] A. Mousavi, M. Rezaee and R. Ayanzadeh, “A survey on compressive sensing: Classical results and recent advancements,” *J. Math. Model.*, vol. 8, no. 3, pp. 309–344, 2020, doi: 10.22124/jmm.2020.16701.1450.

[13] J. Salminen, M. Hopf, S. A. Chowdhury, S. gyo Jung, H. Almerekhi and B. J. Jansen, “Developing an online hate classifier for multiple social media platforms,” *Human-centric Comput. Inf. Sci.*, vol. 10, no. 1, pp. 1–34, Jan. 2020, doi: 10.1186/s13673-019-0205-6.

[14] R. L. Alaoui and E. H. Nfaoui, “Web attacks detection using stacked generalization ensemble for LSTMs and word embedding,” in *Proc. Comput. Sci.*, vol. 215, 2022, pp. 687–696, doi: 10.1016/j.procs.2022.12.070.

[15] D. E. Cahyani and I. Patasik, “Performance comparison of tf-idf and word2vec models for emotion text classification,” *Bull. Electr. Eng. Informatics*, vol. 10, no. 5, pp. 2780–2788, 2021, doi: 10.11591/eei.v10i5.3157.

[16] S. Akuma, T. Lubem and I. T. Adom, “Comparing Bag of Words and TF-IDF with different models for hate speech detection from live tweets,” I*nt. J. Inf. Technol*., vol. 14, no. 7, pp. 3629–3635, 2022, doi: 10.1007/s41870-022-01096-4.

[17] B. C. Öğe and F. Kayaalp, “Farklı Sınıflandırma Algoritmaları ve Metin Temsil Yöntemlerinin Duygu Analizinde Performans Karşılaştırılması”, *DUBİTED,* vol. 9, no. 6, pp. 406–416, 2021, doi: 10.29130/dubited.1015320.

[18] B. Ekici and H. Takcı, “Spam Tespitinde Word2Vec ve TF-IDF Yöntemlerinin Karşılaştırılması ve Başarı Oranının Artırılması Üzerine Bir Çalışma,” *Bilecik Şeyh Edebali Üniversitesi Fen Bilim. Derg.*, vol. 8, no. 2, pp. 646–655, 2021, doi: 10.35193/bseufbd.935247.

[19] K. Koruyan and A. Ekeryılmaz, “Makine Öğrenmesi ile Müşteri Şikayetlerinin Sınıflandırılması,” *AJIT-e Acad. J. Inf. Technol.*, vol. 13, no. 50, pp. 168–183, 2022, doi: 10.5824/ajite.2022.03.004.x.

[20] Ö. Çelik and B. C. Koç, “TF-IDF, Word2vec ve Fasttext Vektör Model Yöntemleri ile Türkçe Haber Metinlerinin Sınıflandırılması”, DEUFMD, vol. 23, no. 67, pp. 121–127, 2021, doi: 10.21205/deufmd.2021236710.

[21] H. Saif, M. Fernandez, Y. He, and H. Alani, “On stopwords, filtering and data sparsity for sentiment analysis of twitter,” in *Proc. 9th Int. Conf. Lang. Resour. Eval. Lr.*, 2014, pp. 810–817.

[22] C. Silva and B. Ribeiro, “The Importance of Stop Word Removal on Recall Values in Text Categorization,” in *Proc. Int. Jt. Conf. Neural Networks*, vol. 3, Aug. 2003, pp. 1661–1666, doi: 10.1109/ijcnn.2003.1223656.

[23] Y. Fan, C. Arora and C. Treude, “Stop Words for Processing Software Engineering Documents: Do they Matter?,” in *Proc. IEEE/ACM 2nd Int. Work. Nat. Lang. Softw. Eng. (NLBSE)*, 2023, pp. 40–47, doi: 10.1109/NLBSE59153.2023.00016.

[24] G. Gupta and S. Malhotra, "Text Document Tokenization for Word Frequency Count using Rapid miner," *Int. J. Comput. Appl.*, vol.12, pp. 24-26, Aug. 2015.
[25] T. Korenius, J. Laurikkala, K. Järvelin and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents," in *Proc. Int. Conf. Inf. Knowl. Manag.*, 2004, pp. 625–633, doi: 10.1145/1031171.1031285.
[26] A. Barbaresi, "Simplemma". *Zenodo, Jan.* 20, 2023. doi: 10.5281/zenodo.7555188.
[27] W. Aljedaani et al., "Sentiment analysis on Twitter data integrating TextBlob and deep learning models: The case of US airline industry," *Knowledge-Based Syst.*, vol. 255, 2022, Art. no. 109780, doi: 10.1016/j.knosys.2022.109780.
[28] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector," in Proc. *1st International Conference on Learning Representations*, 2013, pp.1-12.
[29] A. K. Singh and M. Shashi, "Vectorization of text documents for identifying unifiable news articles," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 7, pp. 305–310, 2019, doi: 10.14569/ijacsa.2019.0100742.
[30] G. Yeşiltaş and T. Güngör, "Intrinsic and Extrinsic Evaluation of Word Embedding Models," in Proc. *Innovations in Intelligent Systems and Applications Conference* (ASYU), 2020, pp. 1-6, doi: 10.1109/ASYU50717.2020.9259855.
[31] D. Jatnika, M. A. Bijaksana and A. A. Suryani, "Word2vec model analysis for semantic similarities in English words," in *Proc. Comput. Sci.*, vol. 157, Sept.. 2019, pp. 160–167, doi: 10.1016/J.PROCS.2019.08.153.
[32] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global vectors for word representation," in Proc. Conference on Empirical Methods in Natural Language Processing, Oct. 2014, pp. 1532–1543. doi: 10.3115/v1/d14-1162.
[33] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967, doi: 10.1109/TIT.1967.1053964.
[34] J. Lakoumentas, J. Drakos, M. Karakantza, G. Sakellaropoulos, V. Megalooikonomou and G. Nikiforidis, "Optimizations of the naïve-Bayes classifier for the prognosis of B-Chronic Lymphocytic Leukemia incorporating flow cytometry data," *Comput. Methods Programs Biomed.*, vol. 108, no. 1, pp. 158–167, 2012, doi: 10.1016/j.cmpb.2012.02.009.
[35] C. Cortes and V. Vapnik, "Support-Vector Networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1023/A:1022627411411.
[36] S. Metlek and K. Kayaalp, "Derin Öğrenme ve Destek Vektör Makineleri İle Görüntüden Cinsiyet Tahmini", DUBİTED, vol. 8, no. 3, pp. 2208–2228, 2020, doi: 10.29130/dubited.707316.
[37] M. Kantardzic, "Decision Trees and Decision Rules," in Data Mining: Concepts, Models, Methods, and Algorithms, 3rd ed. Columbia, MD, U.S.A.: Wiley-IEEE Press, 2019, sec. 6, pp. 197-229
[38] R. Wang, "AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review," in *Proc. Phys. Procedia*, vol. 25, 2012, pp. 800–807, doi: 10.1016/j.phpro.2012.03.160.
[39] E. Scornet, G. Biau and J. P. Vert, "Consistency of random forests," *Ann. Stat.*, vol. 43, no. 4, pp. 1716–1741, Aug. 2015, doi: 10.1214/15-AOS1321.
[40] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: An overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000, doi: 10.1093/bioinformatics/16.5.412.

Conflict of Interest Notice
Authors declare that there is no conflict of interest regarding the publication of this paper**.**

Ethical Approval
It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material
Not applicable.

Plagiarism Statement
This article has been scanned by iThenticate ™.