Available online at sjuoz.uoz.edu.krd

journals.uoz.edu.krd

**Science Journal of University of Zakho**

Vol. 12, No.3, pp.329–335 July-September, 2024

p-ISSN: 2663-628X
e-ISSN: 2663-6298

# AN AUTOMATED NEW APPROACH IN FAST TEXT CLASSIFICATION: A CASE STUDY FOR KURDISH TEXT

Ari M. Saeed*

Department of Computer Science, College of Science, University of Halabja, Halabja, Kurdistan Region, Iraq

ari.said@uoh.edu.iq

Received: 29 Mar., 2024 / Accepted: 25 June., 2024 / Published: 30 July., 2024. https://doi.org/10.25271/sjuoz.2024.12.3.1296

**ABSTRACT:**

With the rapid development of internet technology, text classification has become a vital part of obtaining quick and accurate data. Traditional machine learning methods often suffer from poor performance and high-dimensional feature spaces, which reduce their accuracy. In this paper, the FastText model is proposed as the first-ever classifier on Kurdish text and the results are compared with traditional machine learning methods to show the effects on Kurdish Text. For evaluating the model four datasets Kurdish News Dataset Headlines (KNDH), Medical Kurdish Dataset (MKD), Kurdish-Emotional-Dataset (KMD-77000), and KurdiSent are utilized and compared the results with the traditional machine learning algorithms such as: Random Forest (RF), k-nearest Neighbor (k-NN), Logistic Regression (LR), Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM), Decision Tree (DT), Stochastic Gradient Descent (SGD), as well as the deep learning model Bidirectional Encoder Representations from Transformers (BERT). The outcomes indicate that the FastText model achieved the highest performance with 89% for each precision, recall, F1-score, and 89.10% accuracy for the KNDH dataset. Moreover, when the KMD dataset is utilized the FatText model obtained outperforms all others by approximately 2%. In addition, the comparative analysis showed that FastText is superior when Kurdisent is considered with precision, recall, F1-score, and accuracy by 81.32, 81.83, 81.57, and 81.4 respectively. In addition, when MKD is implemented, the FastText model obtained the highest performance with a precision of 93.32%, recall of 93.36, F1-score of 93.34, and accuracy of 93.1%.

**KEYWORDS**: FastText, Deep Learning, Kurdish Text Classification, Machine Learning, Natural Language Processing

## 1. INTRODUCTION

With the rapid development of technology, the internet has become a necessary part of human life. Various data types are used on the internet, with text being one of the most common forms. Text Classification (TC) is a crucial task in the fields of Natural Language Processing (NLP), Machine Learning (ML), and data mining. Moreover, TC has emerged as a method for analyzing, extracting, detecting, and retrieving user knowledge from large volumes of text. The TC can be categorized into two groups: multi-label and multiclass classification.

In multi-label classification, a text is assigned multiple target labels (such as sports or social), whereas in multiclass classification, a text is assigned only one target label. Some applications of TC include sentiment analysis, news filtering, email sorting, product review categorization, and text message analysis (Umer et al., 2023). Various text classification algorithms, including Support Vector Machine (SVM), Naive Bayes (NB), K-nearest neighbor (KNN), Decision Tree, Neural Network, and FastText, are utilized to classify text. In machine learning algorithms, the Vector Space Model (VSM) is employed to convert text inputs (sentences) into multiple vectors (features) for classification purposes. In contrast to deep learning models, layers are used to prioritize these features (Zulqarnain et al., 2020). The Bag-of-Words (BoW) model and Term Frequency-Inverse Document Frequency (TF-IDF) are implemented to depict the occurrence of words across documents in the corpus. The BoW model transforms documents into vectors, while TF-IDF assigns weights to terms based on their frequency in each document. TF calculates the score for each feature depending on its frequency within each document, whereas IDF determines the count of unique features present in the dataset. In addition, GloVe, BERT, and FastText are word embedding methods utilized to diminish high-dimensional features by considering the similarity between words (Naeem et al., 2022; Singh et al., 2022). GloVe is a pre-trained word embedding model trained on Wikipedia 2014 and Gigaword 5 corpus, and it utilizes a technique that assesses the similarity between word vectors. Meanwhile, Bert, introduced in 2018, a bidirectional pre-trained model designed for various languages. Unlike other models that rely on unidirectional context capture, Bert leverages bidirectional processing (Alammary, 2022).

FastText, developed by Facebook, serves as a library offering pre-trained models for 157 distinct languages, catering to supervised classification and unsupervised text representation tasks (Badri et al., 2022). FastText utilizes character n-grams instead of word n-grams, effectively addressing out-of-vocabulary issues, particularly in languages with complex morphology (Yao et al., 2020). One of the richest languages in terms of morphology is the Kurdish language (Saeed et al., 2023). The significance of this study for the Kurdish language lies in its potential to enhance the accuracy of Kurdish text classification, given the language's intricate morphology. Furthermore, using character n-gram features can improve the efficiency of classifying Kurdish text by offering alternatives for matching out-of-vocabulary words, especially in the presence of typographical errors.

## 2. RELATED WORK

The proliferation of internet applications has led to a massive surge in the volume of text available online. In recent years, automated text mining has become a significant challenge, to extract valuable information from large volumes of content. Automated text classification has driven the creation and enhancement of numerous algorithms for organizing large collections of documents. In a study by Hassan et al. (2022), five

* Corresponding author

machine learning algorithms were applied and compared using two different datasets. The performance of Random Forest (RF), k-nearest Neighbor (k-NN), Logistic Regression (LR), Multinomial Naïve Bayes (MNB), and Support Vector Machine (SVM) was evaluated based on metrics such as accuracy, precision, recall, and F1-score. The results indicated that SVM and LR outperformed the other algorithms on the IMDB English dataset, while k-NN exhibited the best performance on the SPAM dataset. Mokhtar used an Arabic corpus to categorize text into predefined categories such as news, economy, culture, diversity, and sports. In this experiment, five machine learning models, including Random Forest, Logistic Regression, Decision Tree (DT), Stochastic Gradient Descent (SGD), Naïve Bayes (NB), and Support Vector Machine (SVM), were employed. According to the results, Logistic Regression achieved the highest F1 score compared to the other models (Madhfar & Al-Hagery, 2019).

In a recent study, researchers applied machine learning and deep learning algorithms to a medical Kurdish dataset. Before analysis, the text underwent preprocessing steps, such as the removal of irrelevant words and stop words, using the Kurdish Language Processing Toolkit (KLPT) Python library. The study involved a comparison of Bert-multilingual with traditional machine learning algorithms including NB, SGD, DT, Random Forest, SVM, KNN, and LG. The findings indicated that Bert achieved an accuracy of 92%, surpassing the performance of traditional machine-learning algorithms by two percentage points (Badawi, 2023). In another study, researchers addressed the issue of high dimensionality in feature space and the limitations of conventional machine learning algorithms such as SVM, NB, and KNN by introducing FastText as a novel classification model. The study revealed that the FastText model attained an impressive F1-score of 0.9286, outperforming the other models (Yao et al., 2020). Birol Kuyucu used the FastText classifier to analyze the TTC-3600 Turkish dataset. Remarkably, no preprocessing steps such as tokenization, stemming, lemmatization, stop word removal, lowercase conversion, or dimensionality reduction were applied. Following this, the performance of FastText was compared with K-NN, decision tree J48, and Multinomial Naïve Bayes (NV). The results demonstrated that FastText surpassed the other models, achieving an impressive accuracy score of 93.52%, despite the absence of preprocessing steps (Kuyumcu et al., 2019). Additionally, Amalia conducted a comparison between the FastText model and TF-IDF as one of the BOW models for 500 new articles in a low-resource Bahasa Indonesia. The study revealed that TF-IDF requires more preprocessing steps and is time-consuming for model prediction. Moreover, it was observed that FastText classification exhibited superior performance with a 0.97 F1-score compared to TF-IDF (Amalia et al., 2020).

## 3. FASTTEXT ARCHITECTURE

In natural language processing, reduced performance in text classification challenges the accuracy of neural networks. To address this issue, the Facebook research team developed FastText. FastText is a library designed for both supervised and unsupervised learning. Supervised learning can be applied to tasks such as text classification, while unsupervised learning is utilized for learning word embedding from the training dataset. Two primary data sources, Wikipedia and Common Crawl corpus, contribute to collecting the training data for FastText. It is worth noting that FastText offers pre-trained word vectors for 157 different languages, comprising a vast number of vocabulary (600 billion words) from 2 million combined texts, each represented in 300 dimensions (Umer et al., 2023). The structure of FastText closely resembles that of Continuous Bag of Words (CBOW). However, the primary distinction lies in FastText's architecture compared to CBOW, as CBOW utilizes intermediate words rather than labels as shown in Figure. 1

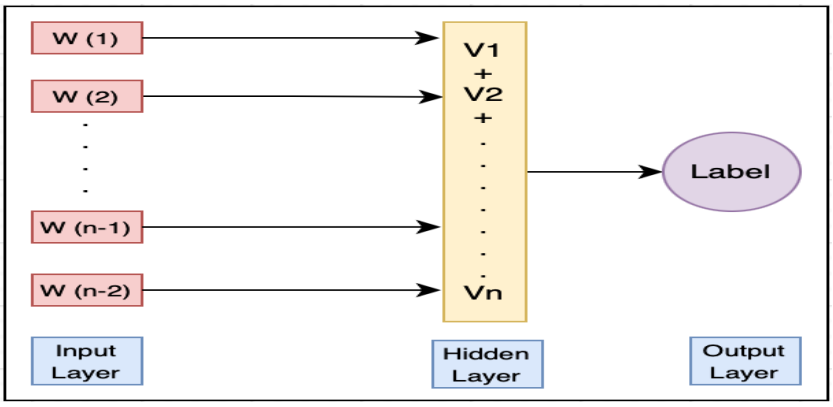


Figure 1: The Basic Framework of the FastText Model

As shown in Figure. 1, the FastText architecture consists of three layers: the input layer, the hidden layer, and the output layer (Dharma et al., 2022).

The input layer of the model processes documents one by one, converting each document into FastText format for utilization within the layer, as illustrated in Table 1:

Tabel 1: Preparing the Documents for Classification with the FastText

| Documents | Labels | Specific Format for FastText Documents |
|---|---|---|
| دکتۆرە گوێز هندی بۆ دە مو چاو باشە | medical | __label__medical دکتۆرە گوێز هندی بۆ دە مو چاو باشە |
| ئەم پیاوە دائیم سەرشۆرە | notmedical | __label__notmedical ئەم پیاوە دائیم سەرشۆرە |

As shown in Table 1, the document is ready for the input layer. After converting the document format, another step is done which is representing words in the documents. The word representation of FastText is different from other models such as word2vec. In word2vec each word is represented as a bag of words while in FastText each word is represented as a bag of character n-gram which is generated vector for unknown words to improve generalization. For example, in character n-gram architecture the word (دکتۆر) is:

<دک, دکت, کتۆ, تۆر, ۆر> when (n-gram=3)

The increase in character n-grams represents a significant improvement over word n-grams and helps address "out of

vocabulary" errors, especially in high-dimensional feature spaces (Khomsah et al., 2022).

The hidden layer, which averages several feature vectors, constructs the Huffman tree. This tree is utilized to determine the most probable function based on the weight and parameters of the class, and it serves output purposes since calculating the tag based on the Huffman coding path can significantly reduce computational load. The SoftMax function is employed in FastText to estimate the likelihood distribution of classes. To define the objective of the model for a dataset containing multiple documents, the following formula (1) is used:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log\left(f(BAX_n)\right) \quad (1)$$

Based on Equation (1), $N$ represents the number of documents in the dataset, $y_n$ denotes the class label for a specific document (the nth document), $f$ signifies the loss function, and $B$ is the weight matrix from the hidden layer to the output layer. Additionally, $A$ represents the weight matrix for word embedding (the embedding layer), and $X_n$ represents the normalized features of the specific document (the nth document). The model employs a linear decay learning rate and stochastic gradient descent for training.

Two important factors have made FastText a robust model: the first is the utilization of the Huffman coding tree-based hierarchical Softmax method, and the second is the utilization of the sub-word n-gram method (Amalia et al., 2020).

## 4. DATASET COLLECTION AND DESCRIPTION

The Kurdish language is a member of the Indo-European language family and is spoken by 40 million people. Kurdish dialects can be broadly categorized into two groups: Sorani and Kurmanji. The Kurdish homeland, known as Kurdistan, spans across four countries: Iraq, Turkey, Iran, and Syria. Both Sorani and Kurmanji are spoken in Iran and Iraq, whereas only Kurmanji is used in Turkey and Syria. Furthermore, Kurds in Iraq and Iran use the Arabic alphabet, while those in Turkey and Syria use the Latin alphabet (Saeed et al., 2018). In this study, four distinct Kurdish datasets were analyzed. These datasets were gathered from various online sources within the Kurdistan region of Iraq (Ahmadi, 2020; Saeed, Ismael, et al., 2022).

The first dataset is the Kurdish News Dataset Headlines (KNDH), compiled from 34 distinct Kurdish channels such as Kurdsat, PayamTV, Rudaw, K24, and others. The proportion of headlines collected from each channel varies. KNDH contains 50,000 headlines categorized into five distinct classes: Health, Science, Economy, Sport, and Social, with each class containing 10,000 headlines. The headlines were gathered using BeautifulSoup and ParsHup software and the texts are labeled automatically (S. Badawi et al., 2023).

The second dataset is the Medical Kurdish Dataset (MKD), which includes 6,756 comments from Facebook. These comments were gathered from various posts related to Education, Sport, Medicine, News, and Economy. After collection, three annotators manually labeled the comments as either medical or non-medical based on their understanding. The dataset was compiled using the Facepager tool (Saeed, Hussein, et al., 2022) The third dataset is the Kurdish-Emotional Dataset (KMD-77000), comprising 77,000 texts collected via the Twitter API. Three annotators proficient in the Kurdish language labeled the texts based on their understanding. The texts were categorized into four classes: joy, sadness, fear, and surprise (S. Badawi, 2023).

The fourth dataset is KurdiSent, comprising 12,000 instances collected from Twitter. After gathering the tweets, three annotators manually labeled them positive, negative, or neutral. To facilitate the annotation process, the open-source text annotation tool Doccano was used for automatic annotation (S. Badawi et al., 2024).

### 1. Implementation and Experiments

The fastText classification model focuses on categorizing Kurdish text documents based on their content. To classify these documents with FastText, several essential steps need to be performed for analyzing and predicting labels, as illustrated in Figure 2:

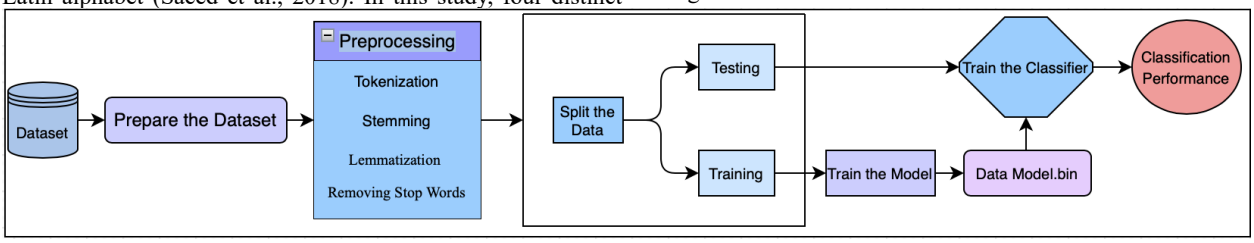


Figure 2: Text Classification Experiment

As demonstrated in Figure 2, when initiating the classification process using FastText, the initial step involves converting a raw dataset into a suitable format for training input. To achieve this, the text documents need to be prepared in a FastText format. This entails prefixing the text with the keyword "__label__", followed by the corresponding class name, such as "medical", and then appending the text document. For instance: "__label__medical this is a good doctor."

The next stage involves preprocessing the text document, a vital step focused on cleaning and readying unstructured textual data for analysis. In this experiment, the Kurdish Language Processing Toolkit (KLPT) is utilized as an open-source Python toolkit for preprocessing, tokenization, stemming, and transliteration (Ahmadi, 2020).

Preprocessing involves a set of techniques applied to tokens to ensure they are clean and consistent. Kurdish has numerous customized keyboard layouts, each assigning different character encodings to visually similar graphemes. Furthermore, keyboards vary in their use for typing Kurdish alongside languages like Persian, Arabic, and Turkish. The Sorani dialect employs various non-uniform graphemes. For example, the grapheme "ی" is represented with five different Unicode characters within the same script: "ي","ي","ى","ي", and "ي","ی"" (with Unicode U+064A, U+0649, U+FEF2, U+FEF1, and U+06CC respectively), with the last one intended for Sorani. To tackle these challenges, two functions, Normalization () and Standardization (), are utilized.

In the Normalization () function, abnormal forms in the text are replaced with standardized forms of the letter (grapheme). The Standardization () function addresses orthographic aspects of the text. Additionally, the unify-numeral () function converts numbers from Persian (۰, ۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹) and Latin (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) forms to Arabic forms (٠, ١, ٢, ٣, ٤, ٥, ٦, ٧, ٨, ٩).

Tokenization involves separating each word in a sentence. While spaces are used to separate words in Arabic script, the process is different in the Kurdish Sorani dialect due to its complex morphology. Not all words correspond directly to tokens. For example, the word "لەخوێندنگاکانماندا" consists of six tokens: "لە","خوێندن","گا","کان","مان","دا". KLPT addresses this complexity

by using an annotated lexicon and a morphological analyzer to tokenize Sorani words. To handle various compound forms, KLPT provides two functions for word tokenization (mwe_tokenize() and word_tokenize()) and uses sent_tokenize() to tokenize sentences based on punctuation.

Transliteration involves converting text from one alphabet to another while retaining the original pronunciation. In the KLPT system, the issue of transliterating characters with dual uses (such as "ی" and "و") in the Sorani dialect has been addressed.

Stemming involves reducing a word to its base form by removing prefixes, affixes, and postfixes. This process is implemented in two classes: Stem and Spellcheck.

The stem class has four functions:

a) Stem (): Retrieves the root of a word, e.g., "بردن" becomes "بر".
b) Lemmatize (): Performs lemmatization, e.g., "بردوویانه" becomes "بردن".
c) Analyze (): Analyzes the morphology of words, returning a dictionary with parts of speech.
d) suffix_suggest (): Returns all possible suffixes that can appear with a given lexeme.

The Spellcheck class includes two functions:

a) check_spelling(): Returns True or False based on the correctness of the spelling.
b) Correct_spelling(): Corrects misspelled words and provides suggestions, e.g., suggesting "بردن" for the misspelled word "بردب".

Following preprocessing, the text document is partitioned for training and testing, employing the holdout method due to the relatively small datasets utilized in this study. The training ratio is set at 80 percent, leaving the remaining 20 percent for testing. Subsequently, the FastText classifier is implemented to train the model, which is then saved as a data model for subsequent steps. Next, the classifier is trained using the testing data alongside the saved data model. The ultimate goal step involves assessing the classification performance.

## 2. Experimental Result

This study evaluates the effectiveness of FastText algorithms on Kurdish text through the utilization of four different Kurdish datasets. The performance of FastText is compared against eight other machine learning and deep learning algorithms. The procedure is structured into several key steps. Initially, the raw datasets are converted into a format compatible with FastText. Subsequently, the data undergoes preprocessing, which includes tokenization, stemming, and the removal of stop words. Afterward, the datasets are split into training and testing sets, with 80% designated for training and 20% for testing purposes. The next phase involves training the model using one of the selected machine learning or deep learning algorithms on the training dataset. Finally, the trained model is evaluated by applying it to the testing dataset to generate predictions.

For evaluating the performance of each classifier, a confusion matrix is used as shown in Table 2:

Tabel 2: Confusion Matrix/ Contingency Table

| | **Positive** | **Negative** |
|---|---|---|
| **Positive** | TP | FN |
| **Negative** | FP | TN |

As illustrated in Table 2, the columns represent the predicted labels, categorized as positive and negative, while the rows denote the actual labels, also categorized as positive and negative. This results in four possible outcomes: True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN).

From Table 2, the following performance metrics are employed to evaluate the results:

Precision: it is used to identify only the relevant data (accurate data) among retrieved data.

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

Recall: it is used to identify all relevant data among retrieved data.

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

F-measure: it is the harmonic mean of precision and recall.

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

Accuracy: indicates the ratio of correctly predicted labels to the total number of predictions

$$Accuracy = \frac{TP + TN}{TP + \mathrm{TN} + \mathrm{FP} + FN} \quad (5)$$

Table 3 evaluates the Precision, Recall, F1-score, and Accuracy for nine algorithms which are implemented on the KNDH dataset as shown below:

Tabel 3: Evaluates the Precision, Recall, F1-score, and Accuracy of Classifiers for the KNDH Dataset

| | **Precision** | **Recall** | **F1** | **Accuracy** |
|---|---|---|---|---|
| **FastText** | 89.00 | 89.00 | 89.00 | 89.10 |
| **NB** | 87.30 | 87.40 | 87.35 | 87.25 |
| **SVM** | 88.01 | 88.91 | 88.46 | 88.53 |
| **DT** | 80.23 | 79.90 | 80.06 | 80.91 |
| **KNN** | 62.80 | 62.30 | 62.55 | 62.36 |
| **LR** | 88.00 | 88.00 | 88.00 | 88.00 |
| **RF** | 85.63 | 85.34 | 85.48 | 85.24 |
| **SGD** | 76.43 | 76.54 | 76.48 | 76.49 |
| **Bert** | 88.26 | 88.29 | 88.27 | 88.12 |

As shown in Table 3, the performance of various classifiers was compared by using precision, recall, F1 score, and accuracy metrics. FastText exhibited the highest values across all metrics, with precision, recall, and F1 scores each at 89.00 and accuracy slightly higher at 89.10. While the SVM, LR, and Bert are all values closest to each other is 88. Moreover, Naive Bayes (NB) achieved a precision of 87.30, recall of 87.40, F1 score of 87.35, and accuracy of 87.25, indicating slightly lower performance compared to SV, LR, and Bert. Random Forest (RF) displayed moderate performance with precision of 85.63, recall of 85.34, F1 score of 85.48, and accuracy of 85.24. DT had lower values, with precision of 80.23, recall of 79.90, F1 score of 80.06, and accuracy of 80.91. SGD and KNN were the least effective, with SGD recording precision of 76.43, recall of 76.54, F1 score of 76.48, and accuracy of 76.49, and KNN showing the lowest performance with precision of 62.80, recall of 62.30, F1 score of 62.55, and accuracy of 62.36. This comparison highlights FastText as the most effective algorithm among those evaluated,

with SVM, LR, and Bert also demonstrating strong performance. indicating its robustness and reliability in text classification tasks.

Table 4 evaluates the Precision, Recall, F1-score, and Accuracy for nine algorithms which are implemented on the KMD dataset as shown below:

Tabel 4: Evaluates the Precision, Recall, F1-score, and Accuracy of Classifiers for the KMD Dataset

| | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **FastText** | 70.31 | 70.39 | 70.35 | 70.5 |
| **NB** | 63.23 | 63 | 63.11 | 63.32 |
| **SVM** | 65.08 | 65.01 | 65.04 | 65.03 |
| **DT** | 63.54 | 63.5 | 63.52 | 63.32 |
| **KNN** | 54.79 | 54.7 | 54.74 | 54.61 |
| **LR** | 65.7 | 65.8 | 65.75 | 65.73 |
| **RF** | 68.44 | 68.48 | 68.46 | 68.47 |
| **SGD** | 44.32 | 44.5 | 44.41 | 44.03 |
| **Bert** | 66.64 | 66.61 | 66.62 | 66.32 |

As shown in Table 4, the KMD dataset was implemented by using various classifiers. It is clear that FastText obtained the highest values across all metrics, with precision, recall, and F1 score each at 70.00 and accuracy slightly higher at 70.5. RF followed closely, achieving a precision of 68.44, recall of 68.48, F1 score of 68.46, and accuracy of 68.46. Bert also performed well, with consistent values of 66.64 across all metrics. Likewise, LR and SVM results are closest to each other with 65 for each precision, recall, F1, and accuracy. In addition, the value of NB is similar to DT for precision, recall, F1, and accuracy, indicating slightly lower performance compared to LR and SVM. Moreover, KNN had lower values, with a precision of 54.79, recall of 54.7, F1 score of 54.74, and accuracy of 54.61. Likewise, SGD was the least effective with a precision of 44.32, recall of 44.45, F1 score of 44.41, and accuracy of 44.03.

Table 5 evaluates the Precision, Recall, F1-score, and Accuracy for nine algorithms which are implemented on the KurdiSent dataset as shown below:

Tabel 5: Evaluates the Precision, Recall, F1-score, and Accuracy of Classifiers for the Kurdisent Dataset

| | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **FastText** | 81.32 | 81.83 | 81.57 | 81.4 |
| **NB** | 75.92 | 75.78 | 75.85 | 75.9 |
| **SVM** | 78.14 | 78.18 | 78.16 | 78.2 |
| **DT** | 73.46 | 73.9 | 73.68 | 73.7 |
| **KNN** | 57.49 | 57.43 | 57.46 | 57.6 |
| **LR** | 79.3 | 79.23 | 79.26 | 79.3 |
| **RF** | 78.09 | 78.12 | 78.10 | 78.12 |
| **SGD** | 71.1 | 71.02 | 71.06 | 71.3 |
| **Bert** | 80.5 | 80.45 | 80.47 | 80.67 |

As shown in Table 5, the results are different for each classification algorithm, it is clear that, when the FastText was implemented, it obtained the highest values across all metrics, 81.32 with precision, 81.83 with recall, and 81.57 with F1 score and 81.4 with accuracy, indicating its robustness and reliability in text classification tasks. Bert followed closely, achieving a precision of 80.5, recall of 88.45, F1 score of 80.47, and accuracy of 80.67. Moreover, LR obtained a high performance of 79 with all metrics. SVM and RF also performed well, with consistent values of 78 across all metrics. In addition, NB showed comparable results with a precision of 75.92, recall of 75.78, F1 score of 75.87, and accuracy of 75.9, making it another strong contender. DT achieved a precision of 73.46, recall of 73.9, F1 score of 73.68, and accuracy of 73.7, indicating slightly lower performance compared to SVM, RF, and NB. SGD displayed moderate performance with precision of 71.1, recall of 71.02 F1 score of 71.06, and accuracy of 71.3. KNN showed the lowest performance with precision of 57.49, recall of 57.43, F1 score of 57.46, and accuracy of 57.6.

Table 6 evaluates the Precision, Recall, F1-score, and Accuracy for nine algorithms which are implemented on the MKD dataset as shown below:

Tabel 6: Evaluates the Precision, Recall, F1-score, and Accuracy of Classifiers for the MKD Dataset

| | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| **FastText** | 93.32 | 93.36 | 93.34 | 93.1 |
| **NB** | 92.9 | 92.93 | 92.91 | 92.8 |
| **SVM** | 93.1 | 93.3 | 93.20 | 93.3 |
| **DT** | 87.68 | 87.64 | 87.66 | 87.66 |
| **KNN** | 62.34 | 62.1 | 62.22 | 62.31 |
| **LR** | 90.5 | 90.51 | 90.50 | 90.49 |
| **RF** | 91.31 | 91.2 | 91.25 | 91.3 |
| **SGD** | 63.31 | 63.33 | 63.32 | 63.45 |
| **Bert** | 92.1 | 92 | 92.05 | 92.01 |

As shown in Table 6, the performance of various text classification algorithms was compared using precision, recall, F1 score, and accuracy metrics. FastText exhibited the highest values across all metrics, with 93.32 precision, 93.36 recall, and 93.34 F1 score and accuracy at 93.1, indicating its robustness and reliability in text classification tasks. Support Vector Machine (SVM) followed closely, achieving a precision of 93.1, recall of 93.3, F1 score of 93.2, and accuracy of 93.3. NB also performed well, with consistent values of 92.9 across all metrics. Bert showed comparable results with a precision of 92.1, recall of 92, F1 score of 92.05, and accuracy of 92.01, making it another strong contender. In addition, RF achieved a precision of 91.31, recall of 91.2, F1 score of 91.25, and accuracy of 91.3, indicating slightly lower performance than Bert and SVM. LR displayed

moderate performance with precision of 90.5, recall of 90.51, F1 score of 90.5, and accuracy of 90.49. DT had lower values, with precision of 87.68, recall of 87.64, F1 score of 87.66, and accuracy of 87.66. SGD and KNN were the least effective, with SGD recording precision of 63.31, recall of 63.33, F1 score of 63,32, and accuracy of 63.45, and KNN showing the lowest performance with precision of 62.34, recall of 62.1, F1 score of 62.22, and accuracy of 62.31.

Another important metric for evaluating the FastText classifier is calculating space and time complexity for each dataset as shown below 7:

Tabel 7: Space and Time Complexity of FastTexl Model

| | Training sample | Vocabulary size | Training time | Inference time |
|---|---|---|---|---|
| **Medical** | 4729 | 20168 | 0.307 | 0.078 |
| **Kurdisent** | 8614 | 22079 | 0.375 | 0.113 |
| **KNDH** | 35000 | 43427 | 0.964 | 0.594 |
| **KMD** | 54089 | 53400 | 1.646 | 0.900 |

As shown in Table 7, the FastText model is implemented on four distinct datasets, Medical, Kurdisent, KNDH, and KMD, and reveals noteworthy differences in their performance metrics. The Medical dataset is the smallest with 4,729 samples and a vocabulary size of 20,168, and shows improved efficiency, with a training time of 0.307 seconds and an inference time of 0.078 seconds. Kurdisent, with 8,614 training samples and a vocabulary size of 22,079, shows improved efficiency, with a training time of 0.375 seconds and an inference time of 0.113 seconds. KNDH, considerably larger with 35,000 samples and a vocabulary size of 43,427, demonstrates increased computational demand, requiring 0.964 seconds for training and 0.594 seconds for inference. The largest KMD dataset with 54,089 samples and a vocabulary size of 53,400, has the highest training and inference times of 1.646 and 0.900 seconds, respectively. These results indicate a clear trend: as the dataset size and vocabulary increase, both training and inference times rise significantly, highlighting the scalability challenges of the FastText model.

It can also be concluded that the FastText classifier outperforms all other classifiers in classifying Kurdish text. This effectiveness of FastText is due to two main factors. Firstly, it employs a high-quality n-gram character approach to mitigate out-of-vocabulary errors. Secondly, it uses the hierarchical Softmax method based on the Huffman coding tree as the final layer in training neural networks.

## CONCLUSIONS

The primary objective of this study is to introduce a FastText-based classifier for classifying Kurdish language text. The proposed method was compared with eight traditional machine learning and deep learning algorithms. To assess the performance of each classifier, four Kurdish datasets were used. After preprocessing steps such as tokenization, stemming, lemmatization, and stop word removal, the datasets were divided into training and testing sets. The study demonstrated that FastText achieved the highest precision, recall, F1 score, and accuracy compared to all other algorithms across all datasets. Based on these findings, it can be concluded that FastText is the most effective classifier for Kurdish language text classification. Future research can expand on this method to develop a hybrid model based on the Fasttext model for more efficient text classification in the Kurdish language.

## REFERENCES


Ahmadi, S. (2020). KLPT–Kurdish language processing toolkit. *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, 72–84.

Alammary, A. S. (2022). BERT Models for Arabic Text Classification: A Systematic Review. *Applied Sciences 2022, Vol. 12, Page 5720*, *12*(11), 5720. https://doi.org/10.3390/APP12115720

Amalia, A., Sitompul, O. S., Nababan, E. B., & Mantoro, T. (2020). An Efficient Text Classification Using fastText for Bahasa Indonesia Documents Classification. *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics, DATABIA 2020 - Proceedings*, 69–75. https://doi.org/10.1109/DATABIA50434.2020.9190447

Badawi, S. (2023). KMD: A New Kurdish Multilabel Emotional Dataset For the Kurdish Sorani Dialect. In M. Abbas & A. A. Freihat (Eds.), *Proceedings of the 6th International Conference on Natural Language and Speech Processing (ICNLSP 2023)* (pp. 308–315). Association for Computational Linguistics. https://aclanthology.org/2023.icnlsp-1.33

Badawi, S., Kazemi, A., & Rezaie, V. (2024). KurdiSent: a corpus for kurdish sentiment analysis. *Language ResourcesandEvaluation*,1–20. https://doi.org/10.1007/S10579-023-09716-6/METRICS

Badawi, S. S. (2023). Using Multilingual Bidirectional Encoder Representations from Transformers on Medical Corpus for Kurdish Text Classification. *ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY*, *11*(1), 10–15. https://doi.org/10.14500/aro.11088

Badawi, S., Saeed, A. M., Ahmed, S. A., Abdalla, P. A., & Hassan, D. A. (2023). Kurdish News Dataset Headlines (KNDH) through multiclass classification. *Data in Brief*, *48*, 109120. https://doi.org/10.1016/j.dib.2023.109120

Dharma, E. M., Gaol, F. L., Leslie, H., Warnars, H. S., & Soewito, B. (2022). THE ACCURACY COMPARISON AMONG WORD2VEC, GLOVE, AND FASTTEXT TOWARDS CONVOLUTION NEURAL NETWORK (CNN) TEXT CLASSIFICATION. *Journal of Theoretical and Applied Information Technology*, *31*(2). www.jatit.org

Khomsah, S., Ramadhani, R. D., & Wijaya, S. (2022). The Accuracy Comparison Between Word2Vec and FastText On Sentiment Analysis of Hotel Reviews. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *6*(3), 352–358. https://doi.org/10.29207/resti.v6i3.3711

Kuyumcu, B., Aksakalli, C., & Delil, S. (2019). An automated new approach in fast text classification (fastText): A case study for Turkish text classification without pre-processing. *ACM International Conference Proceeding Series*, 1–4. https://doi.org/10.1145/3342827.3342828

Naeem, M. Z., Rustam, F., Mehmood, A., Mui-zzud-din, Ashraf, I., & Choi, G. S. (2022). Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms. *PeerJ Computer Science*, *8*, e914.https://doi.org/10.7717/PEERJ-CS.914/SUPP-4

Saeed, A. M., Badawi, S., Ahmed, S. A., & Hassan, D. A. (2023). Comparison of feature selection methods in Kurdish text classification. *Iran Journal of Computer Science*, 1–10.

Saeed, A. M., Hussein, S. R., Ali, C. M., & Rashid, T. A. (2022). Medical dataset classification for Kurdish short text over social media. *Data in Brief*, *42*, 108089. https://doi.org/10.1016/J.DIB.2022.108089

Saeed, A. M., Ismael, A. N., Rasul, D. L., Majeed, R. S., & Rashid, T. A. (2022). *Hate Speech Detection in Social Media for the Kurdish Language*. 253–260. https://doi.org/10.1007/978-3-031-14054-9_24

Saeed, A. M., Rashid, T. A., Mustafa, A. M., Agha, R. A. A.-R., Shamsaldin, A. S., & Al-Salihi, N. K. (2018). An evaluation of Reber stemmer with longest match stemmer technique in Kurdish Sorani text classification. *Iran Journal of Computer Science*, *1*(2), 99–107. https://doi.org/10.1007/s42044-018-0007-4

Singh, K. N., Devi, S. D., Devi, H. M., & Mahanta, A. K. (2022). A novel approach for dimension reduction using word embedding: An enhanced text classification approach. *International Journal of Information Management Data Insights*, *2*(1). https://doi.org/10.1016/j.jjimei.2022.100061

Umer, M., Imtiaz, Z., Ahmad, M., Nappi, M., Medaglia, C., Choi, G. S., & Mehmood, A. (2023). Impact of convolutional neural network and FastText embedding on text classification. *Multimedia Tools and Applications*, *82*(4), 5569–5585. https://doi.org/10.1007/s11042-022-13459-x

Yao, T., Zhai, Z., & Gao, B. (2020). Text Classification Model Based on fastText. *Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Information Systems, ICAIIS 2020*,154–157. https://doi.org/10.1109/ICAIIS49377.2020.9194939

Zulqarnain, M., Ghazali, R., Mazwin, Y., Hassim, M., & Rehan, M. (2020). A comparative review on deep learning models for text classification. *Indonesian Journal of Electrical Engineering and Computer Science*,*19*(1),325–335. https://doi.org/10.11591/ijeecs.v19.i1.pp325-335