Sakarya Üniversitesi Fen Bilimleri Dergisi
Sakarya University Journal of Science

ISSN : 2147-835X
Publisher : Sakarya University

Vol. 29, No. 1, 1-17, 2025
DOI: https://doi.org/10.16984/saufenbilder.1349956

Research Article

Offensive Language Detection in Turkish Language by Using NLP

Bekir Furkan Kesgin*, Rüştü Murat Demirer

Bahcesehir University, Faculty of Engineering, Department of Computer Engineering, Istanbul, Türkiye, furkan_bfk@hotmail.com, rustumurat.demirer@ou.bau.edu.tr
*Corresponding Author

ARTICLE INFO

Keywords:
Cyberhate
Social media
Natural language processing
Classification algorithms
Cyberbullying

Article History:
Received: 25.08.2023
Revised: 12.01.2025
Accepted: 12.01.2025
Online Available: 12.02.2025

ABSTRACT

The growing use of social media has increased online harassment, cyberhate, and the use of offensive language. This poses significant challenges for effectively detecting and addressing such issues. Natural Language Processing (NLP) has seen considerable advancements; however, automatically identifying offensive language remains a complex task due to the ambiguous and informal nature of user-generated content and the social context in which it occurs. In this thesis, our goal is to develop methods for automatic detection of offensive language in social media. Multiple classification algorithms, including Multinomial Naive Bayes, Gaussian Naive Bayes, SVM, Logistic Regression, and LSTM, are implemented and evaluated. Key measures including accuracy, F1 score, and AUC score are used to evaluate how well these algorithms work. Results show that the Random Forest Classifier obtains an AUC score of 0.65 and an accuracy of 0.82 without word2vec. On the other hand, LSTM demonstrates a competitive AUC score of 0.78 when compared to the Random Forest Classifier. These findings provide insights into the effectiveness of different algorithms for offensive language detection. The research contributes to the field by providing valuable tools and insights to enhance Turkish language processing and prioritize online safety, particularly in combating cyberbullying and fostering a tolerant online environment. The findings also pave the way for future research endeavors in natural language processing and have practical implications for protecting individuals and promoting a secure online space.

1. Introduction

The power of language has played a crucial role in human evolution, enabling communication, fostering development, and driving progress. One significant advancement in language processing is the Enigma technology, which emerged during World War II to decode enemy messages. Initially, there were concerns among computer scientists regarding the outcomes of natural language processing (NLP). Some researchers believed in the progress through statistics and probability, while others emphasized the importance of predefined rules for computers [1]

of expression but also reinforces prejudices, fosters discrimination, and incites violence. Many platforms have community standards and reporting mechanisms, but additional measures may be needed to ensure the safety and well-being of users [3].

Addressing hate speech poses challenges in languages other than English due to the lack of readily available data models. While English has been extensively researched and has abundant data, Turkish, for example, faces a scarcity of data. Creating data for Turkish is crucial to identify and minimize the psychological impacts of improper language, especially on young individuals [2].

The anonymity of cyberspace has allowed people to express themselves freely, but it has also led to the spread of hateful and discriminatory messages. Hate speech not only violates freedom

Cite as: B. F. Kesgin, R. M. Demirer (2025). Offensive Language Detection in Turkish Language by Using NLP, *Sakarya University Journal of Science*, 29(1), 1-17. https://doi.org/10.16984/saufenbilder.1349956

Hate speech can exacerbate mental health issues and pose risks to individuals who may be susceptible to negative actions. Identifying and removing hate speech is therefore vital for people's well-being [4].

With the increasing use of social media, cyberbullying has become a prominent issue.

However, research on cyberbullying in languages other than English, such as Turkish, is limited. This restriction hampers comprehensive studies and solutions for Turkish-speaking individuals.

While the right to free expression permits the use of abusive language on social media, this situation is untenable. Developing automated solutions is necessary to filter the growing amount of content and mitigate the negative consequences, especially for young users.

Language differences, such as the use of suffixes in Turkish and prefixes in English, have implications for word creation and meaning [5].

In summary, language plays a significant role in human progress, but challenges remain in addressing hate speech and cyberbullying. Further research, data availability, and automated solutions are needed to promote a safer and more inclusive online environment.

The research contributes to following items;

1. Collecting offensive words from the internet and processing input data for stemming, suffixes, censoring, and various options.
2. Collection of numerous data from famous internet websites by using the words that are generated.
3. Making encoding to make appropriate for deep learning models.
4. To classify items, GaussianNB, MultinomialNB, Logistic Regression, XGBClassifier, LSTM, SVC, etc. are used.
5. Results are shown by accuracy, f-1 score with confusion matrix on the offensive text of Turkish language.
6. This research helps Turkish language processing and help with making application who have kids. Because cyberbullying in the Turkish language is not good enough to detect it.

2. Neural Network Architectures Implemented in Natural Language Processing

Natural language processing may be defined as the modeling of rule-based human language and its transmission to a computer. It is a machine learning technique that can understand and comprehend human language. Artificial neural networks include NLP as a subclass. It provides us with a sophisticated language processing approach that blends machine learning with deep learning. Linguists put a lot of work into training models since natural language processing involves more than simply code and incorporates a lot of information about humans. It is one of the functions in the background of programs that may recognize real-world voice instructions and transform them to text.

As we looked at the encodings available in machine learning, we discovered that word2vec and one hot encoding were the best fit for our data set. According to Ma and Zhang (2015), utilizing Word2Vec on huge data is preferable due to performance and the utilization of NLP regions [6].

2.1. Encoding

In NLP, it is necessary to convert textual data into a numerical format for machine learning models to process it. This conversion process is called encoding and can be performed at either the word or character level. The encoding method used is important for effective data processing and analysis.

2.2. Word2Vec

Word2Vec works by extracting words from a phrase one at a time and assigning numbers to the most frequently used ones. It is an encoding strategy that establishes a context inside itself by examining the words to the right and left of the term where it is utilized semantically (Mikolov, T., Chen, K., Corrado, G. S., Dean, J., 2013). Figure 1 shows the semantic links between word pairs like king and queen or man and woman,

demonstrating Word2Vec's capacity to capture analogical relationships.

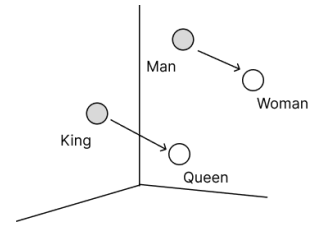

Figure 1. Word2vec example

In Word2Vec, the vector size determines the amount of features used to represent a word. As the number of features increases, so does the vector's size. A larger vector size allows for more complex word representations, but it requires more computer power and data. In contrast, a lower vector size produces more basic word representations with fewer information, but it requires less processing power and data.

Word2Vec is a technique used in natural language processing that represents words with numerical vectors. These vectors contain attributes that indicate the meaning of a word. Figure 2 shows how Word2Vec may capture geographical associations such as nations and capitals. These examples demonstrate how Word2Vec learns word associations and similarities, making it ideal for a wide range of NLP applications [7].

Figure 2. Example of word2vec

2.2.1. Index based encoding

Index-based encoding is an encoding method that helps to make categorical data more meaningful to a model. Unlike word2vec, it adds data to the vector space within itself, instead of defining the data within a one-dimensional object. It does this by encoding all values with values as 0 and 1. Index-based encoding is a method used in NLP to encode a word or text by connecting each word to a word index 6 in the dictionary. This approach is more memory-efficient and performs faster processing, depending on the size of the vocabulary.

In mathematical terms, index-based encoding considers the size of the dictionary as n and assigns a numerical index to each word, with the index number being a whole number between 0 and n-1. For instance, if a dictionary has 10,000 words, each word will be assigned a sequential index number ranging from 0 to 9,999.

Compared to other encoding methods like one-hot encoding or binary encoding, index-based encoding provides a smaller representation for word vectors. The data is pre-processed before beginning the real analysis in order to make algorithms easier to use and increase efficiency [8]. Figure 3 provides a practical example of index-based encoding, where each word in a dictionary is assigned a unique numerical index, demonstrating its memory-efficient representation.

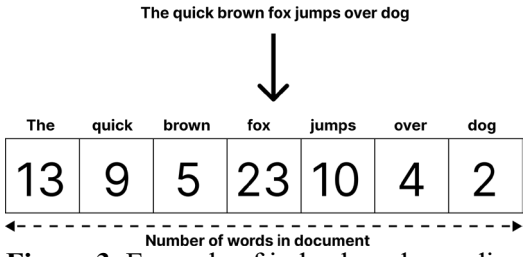

Figure 3. Example of index based encoding

Mathematic of Classifiers

The classifier can be represented mathematically as a formula in $\mathcal{D}$ space:
$S_n = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ in $\mathcal{X} \times \mathcal{Y}$ space relevant to unsupervised learning approach. Because this distribution is unknown.
$\mathbf{x}$ are word2vec or index based encoded vectors. They are subspace of $\mathcal{X} \subseteq [0,1]^d$ and $\mathcal{Y} = \{y_1, y_2, y_3, \ldots\}$ label space we focus on binary classification ($\tau = 2$) to obtain classification error.

$$R_{\mathcal{D}}(h) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq y]$$
$$= E_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbb{I}[h(\mathbf{x}) \neq y]\right]$$
$$= E_{\mathbf{x}\sim\mathcal{D}_{\mathcal{X}}}\left[\sum_{j=1}^{\tau} \eta_j(\mathbf{x})\mathbb{I}[h(\mathbf{x}) \neq j]\right]$$

$$R^*_{\mathcal{D}} = E_{\mathbf{x}}\left[\min_{j\in[\tau]}\{1-\eta_j(\mathbf{x})\}\right] \text{ and } h^*_{\mathcal{D}}(\mathbf{x}) = \arg\max_{j\in[\tau]}\{\eta_j(\mathbf{x})\}$$

$R^*_{\mathcal{D}}$ is the minimum set of the classification. Where probability condition is

$\eta j(x) = Pr[y = j|x]$ for $j \in [\tau]$ of $y = j$ over offensive or non-offensive $\mathbf{x}$ with distribution of $\mathcal{D}$, and $\sum_{j=1}^{\tau=2} \eta_j(\mathbf{x}) = 1$ E is the expectation error in this formula and $E_{\mathbf{x}}$ this get the average of data inside the bracket. $R_{\mathcal{D}}(h)$ over whole training of this size. The function $\mathbb{I}[\cdot]$ is referred to as the indicator function, and it outputs 1 when the statement it evaluates will be true and 0 when it's false.

$h(\mathbf{x})$ is the hypothesis whether word offensive or non-offensive.

$\mathcal{D}$ is the training dataset which includes both offensive – non-offensive corpus data and unknown in practice [9].

2.3. Decision trees

Decision trees are a common technique in natural language processing (NLP) for tasks including sentiment analysis, text categorization, and named entity identification [10]. A decision tree is a type of model made up of leaf nodes, branches, and internal nodes. The internal nodes represent the feature tests, the branches of the results of the tests, and the leaf nodes the projected values or class labels. Figure 4 illustrates an example of a decision tree structure, showing how internal nodes split data based on feature tests, with branches leading to leaf nodes that represent class labels or outcomes.

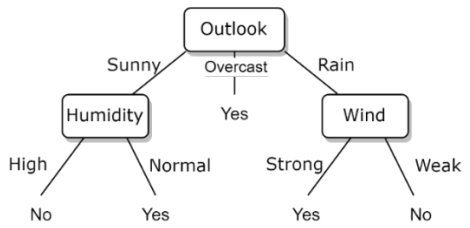

Figure 4. Example decision trees structure

Decision trees can be created using different methods to extract characteristics from the data. The algorithm then learns the most effective divisions on these features, aiming to maximize information gain or reduce entropy at each node. Nevertheless, decision trees have the tendency to overfit when dealing with intricate or noisy data. To tackle this problem, pruning techniques like reduced error pruning and cost-complexity pruning can be utilized. These methods aim to simplify the tree structure by eliminating superfluous nodes or branches that do not significantly contribute to the overall performance. Ensemble methods like random forests and gradient boosting can improve decision tree accuracy and robustness by combining multiple trees.

2.3.1. Random forest classifier

Using random forest in the context of big data provides convenience and consistency in estimating large datasets. It reduces the risk of overfitting and is less likely to suffer from this issue due to the combination of decision trees. However, it also comes with challenges, such as a complex structure formed by multiple decision trees and increased storage requirements.

Another machine learning model, the Extra Trees Classifier, offers faster performance and consumes fewer computational resources by randomly selecting features [11]. However, it performs worse than the Random Forest Classifier. Both models are ensemble methods that combine decision trees and utilize random feature selection, improving prediction accuracy. The Extra Trees Classifier is more efficient and requires fewer resources, while the Random Forest Classifier is more effective but takes longer to train.

Figure 5 shows an example of the Random Forest algorithm's structure and prediction process when applied to huge datasets, which helps to better understand its performance.

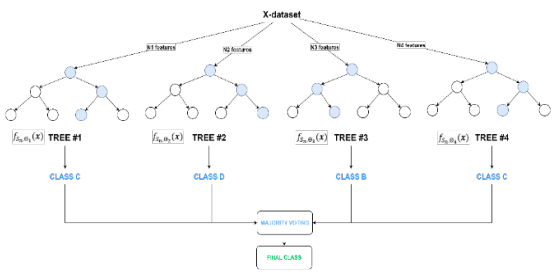

Figure 5. Example random forest classifier

The random forest classifier, $f_m(x)$, aggregates the outputs of m individual randomized trees, $f_{S_n,\Theta_1}(x)$, $f_{S_n,\Theta_2}(x)$, $f_{S_n,\Theta_3}(x)$, $\ldots . f_{S_n,\Theta_m}(x)$ by taking a majority vote. Random Forest Classifier is that;

$$f_m(\mathbf{x}) = \arg\max_{j\in[\tau]} \left\{ \sum_{i=1}^{m} \mathbb{I}\left[f_{S_n,\Theta_i}(\mathbf{x}) = j \right] \right\}$$

The function $f_m(\mathbf{x})$ selects the maximum value j among all possible values in the set $[\tau]$, based on the indicator function that evaluates the condition $f_{S_n,\Theta_i}(\mathbf{x}) = j$, where m and i range from 1 to m value and the ties are broken arbitrarily. The vectors Θ_1, Θ_2,..., Θ_m are randomly distributed and independent of each other, and they define the process of selecting the split leaves, dimensions, and positions when constructing randomized trees. In the following sections, the specific vectors, $\Theta_1, \Theta_2, \ldots \Theta_m$ will be defined for different random forests.

In simpler terms, this formula describes how the Random Forest classifier works by using multiple decision trees to classify data points and aggregating the predictions based on the most common result.

2.4. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning technique used for classification and regression [12]. SVM may be used to categorize text as relevant or irrelevant in the context of NLP text detection by considering particular properties.

The fundamental principle behind SVM is to find a decision boundary that maximizes the margin between two classes. The margin is the distance between the decision border and the support vectors, or nearest data points, for each class.

To determine this decision boundary, SVM utilizes an optimization problem-solving approach that aims to maximize the margin while adhering to specific constraints. SVM seeks to minimize the following objective function:

$$(1/2)\left|\left|w\right|\right|^2 + C\sum_i \xi_i$$

In this context, the weight vector is denoted as w, and ξ_i represents the slack variable associated with each data point. The regularization parameter C regulates the trade-off between maximizing the margin and reducing the classification error. The objective function seeks to find the optimal value of w and ξ_i that minimizes the classification error subject to the constraints:

$$y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Here, y_i is label of the class for the i-th data point, and the bias term is b. Optimal value of w and b are found, the decision boundary is given by: $w^T x + b = 0$ The classification decision is then made based on the sign of $w^T x + b$ or $k(x, x') = \exp(-\|x - x'\|^2/2\sigma^2)$ may be used to map data into higher dimensions for non-linear classification tasks.

Figure 6 demonstrates an example of SVM decision boundaries, showcasing the optimal margin, support vectors, and separation of two classes in a two-dimensional space.

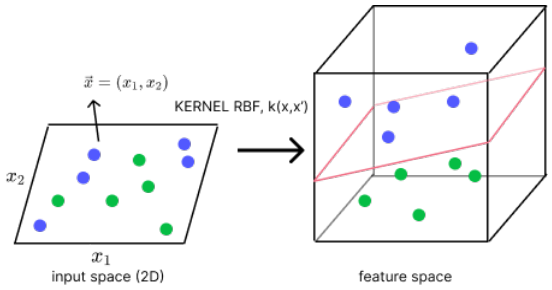

Figure 6. SVM example

2.4.1. Radial Basis Function (RBF)

The SVM-RBF algorithm is a machine learning technique that classifies data by mapping it to a feature space and measuring the distance between points in that space to represent the data in a higher dimensional space. In a Word2vec model with SVM-RBF, the first step is to learn word embeddings and then represent them in the vector space of the feature space. Subsequently, the SVM-RBF algorithm uses these representation of vector in the feature space to classify the data [13]. RBF function is like below;

$$k(x,y) = \exp\left(-\frac{\| x - y \|^2}{2\sigma^2}\right)$$

The performance of the SVM-RBF classifier under varying regularization parameters (C) is demonstrated in Figure 7.

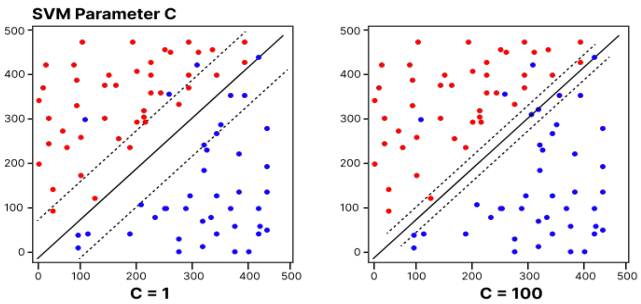

Figure 7. SVM Classifier with different C parameters

2.5. Long Short-Term Memory (LSTM)

LSTM networks are utilized in natural language processing (NLP) applications such as sentiment analysis, language modeling, and machine translation. By considering the prior context, language models may calculate the likelihood of a word sequence in the context of language modeling. Long short-term memory (LSTM) and word2vec have become a common pipeline for jobs in natural language processing [14]. The pipeline has several stages:

2.5.1. Data preprocessing

Preprocessing, lower casing, and punctuating the raw input data into training, validation, and testing sets.

2.5.2. Word embedding

The text is turned into numbers using word embeddings such as word2vec. This makes it easy to use with an LSTM.

2.5.3. LSTM network architecture

The next step is to design an LSTM network architecture, as illustrated in Figure 8, which depicts the integration of one or more LSTM layers, a dropout layer for regularization, and a final dense layer for output classification. This figure provides a schematic representation of how LSTM layers consume word embeddings as input, utilizing memory cells to capture context and long-term dependencies between words in the text. Additionally, the dropout layer, shown in the diagram, mitigates overfitting by selectively deactivating neurons during training. Finally, the dense layer translates the processed information into class probabilities, facilitating output classification.

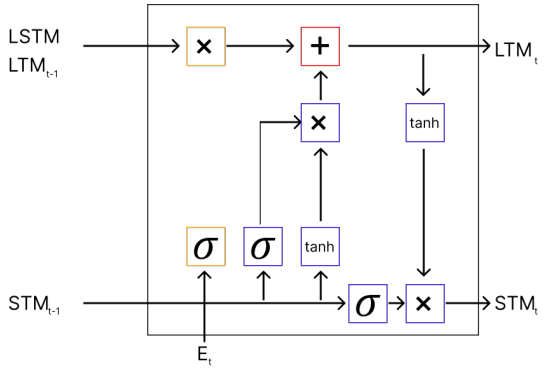

Figure 8. LSTM model working schema

2.5.4 Model training

The LSTM network is trained in cleaned, z-scored data. It has two types of cross-entropy loss functions, one for binary classification and one for multi-class classification.

2.5.5. Model evaluation

After training, the LSTM network is tested on the validation and test sets to see if it works on new data. Its performance is measured by metrics including accuracy, precision, recall, and F1-score.

2.5.6. Model deployment

LSTM can infer user-generated text inputs or real-time data streams after training. This pipeline uses word2vec and LSTM networks for tasks involving natural language, such as named entity identification, text categorization, and sentiment analysis. These components are instrumental in processing and analyzing textual data, facilitating the execution of these language-oriented tasks.

2.5.7. Optimization

Due to the complexity of these models and the requirement for processing huge volumes of textual input, LSTM models in NLP text

classification applications require optimization. By effectively updating the model's parameters and minimizing the loss function, optimization techniques enable the model to learn from the data and enhance its performance and accuracy. Additionally, by modifying the model's weights and biases during training to enhance the prediction performance of minority classes, optimization approaches can assist solve the problem of unbalanced datasets. To achieve high accuracy and performance in text classification tasks using LSTM models in NLP, optimizations are therefore essential [15].

LSTM has optimizations like below;

2.5.7.1. Gradient descent

It is used to minimize loss. Making good predictions is important, so we need to change model parameters. The most common way to do this is "mass gradient descent." This method uses the data set and updates the parameters by computing all the data at each step. This method is faster than other approaches but can produce poorer results if the data is not large enough.

2.5.7.2. Cross-entropy loss

Error measurement is done using the cross-entropy loss. For given review, its predicted sentiment is y and its actual sentiment is y. $L(y,\hat{y}) = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$ The goal of gradient descent is to minimize the average cross-entropy loss over the entire training set and the average loss over the training set as $J(\theta)$. Then the gradient descent rule for θ is:

$$\theta = \theta - \alpha\frac{\partial J(\theta)}{\partial\theta}$$

where α is the learning rate, which controls the step size of each update. The derivative of $J(\theta)$ with respect to θ can be computed using the chain rule of calculus:

$$\frac{\partial J(\theta)}{\partial\theta} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L(y_i,\hat{y_i})}{\partial\hat{y_i}}\frac{\partial\hat{y_i}}{\partial\theta}$$

where N is the total number of training instances, and y_i and $\hat{y_i}$ are the actual and predicted sentiments for the i-th training example.

2.5.7.3. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a widely using optimization algorithm for updating the parameters $\boldsymbol{\theta}$ of a Deep Neural Network (DNN). In contrast to regular gradient descent, which computes on the entire dataset, SGD randomly selects a small batch of the dataset and performs computations on it. This makes SGD more efficient and capable of producing similar performance as regular gradient descent when the learning rate $\boldsymbol{\eta}$ is low. In LSTM networks, SGD updates the weight vectors iteratively by calculating the gradient from a randomly selected sample.

The goal of SGD is to minimize the loss function $L(\theta, D)$ given the dataset D, where θ is the parameters set. At each step, SGD updates θ with one step towards the negative gradient as follows:

$\theta_{t+1} = \theta_t - \eta\nabla_\theta L(\theta_t, x_i, y_i)$, which is the update formula used by SGD to update the parameters θ at each step.

where $\nabla_\theta L(\theta_t, x_i, y_i)$ corresponds to the gradient of the loss function L with respect to the current parameters θ_t and the randomly selected sample (x_i, y_i).

2.5.7.4. Adaptive Moment Estimation (ADAM)

This can train an LSTM network with parameters $\boldsymbol{\theta}$ using the cross-entropy loss function. $\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\alpha}\frac{\hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{v}}}+\boldsymbol{\epsilon}}$ where α,$\hat{m}$ and $\hat{v}$ are learning rate, first and second gradients, and small constant $\boldsymbol{\epsilon}$.

$\hat{m}t = \beta_1\hat{m}t - 1 + (1-\beta_1)g_t$
$\hat{v}t = \beta_2\hat{v}t - 1 + (1-\beta_2)g_t^2$

where t is the current time step, g_t is the loss's gradient with respect to parameters at time step t , and $\beta_1$and β_2 are the first and second moment exponential decay rates.

ADAM combines momentum and adaptive learning rate methods. The momentum term speeds up and stops oscillations, while the adaptive learning rate term adjusts the step size. We do this until the loss is minimal.

The best optimization strategy for LSTM training in NLP depends on the task and dataset. Try different strategies to find the one that works best.

2.6. Gausian naive bayes

Gaussian Naive Bayes is a popular classification technique that employs Bayes theorem to classify data. Bayes' theorem provides a framework for calculating the an event's likelihood depending on the available information about the conditions that influenced the occurrence of the event [16].

This theorem is applied in classification problems to identify the class to which a particular data point belongs. The "Naive" in Gaussian Naive Bayes refers to the assumption that each feature is not dependent on the other features. The "Gaussian" assumption is based on the idea that features follow a normal distribution, making the algorithm effective when data follows this pattern. The algorithm works by calculating the probability of each feature independently and then combining these probabilities to determine the data point's class. Gaussian Naive Bayes is a straightforward and powerful method for classifying data.

$$P(y|x_1, x_2, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{\sum_{y'} P(y')\prod_{i=1}^{n} P(x_i|y')}$$

In this equation:

y represents the class. $x_1, x_2, \dots, x_n$ represent the features. $P(y|x_1, x_2, \dots, x_n)$ represents the probability of the class given the features. $P(y)$ indicates the class's previous probability. $P(x_i|y)$ represents the conditional probability of feature x_i in a given class y where

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

x is the random variable. μ represents the mean, σ^2 represents the variance. In other words, the Gaussian probability density function is a way to calculate the probability of x value occurring in a normal distribution with a given mean and variance. It is used in various statistical analyses, including Gaussian Naive Bayes classification.

2.7. Multinominal naive bayes

Multinomial Naive Bayes algorithm is designed to be used in problems such as text classification [17]. This algorithm is used to determine which category a document belongs to. For example, classifying an email as spam or non-spam. This algorithm calculates the frequency of each word in the document and uses these frequencies to make probability calculations. By ignoring the dependencies between words and treating each word separately, the algorithm calculates the probability of each word and multiplies them to determine the class of the document.

$$P(c_j|d) = \frac{\prod_{i=1}^{n} P(t_i|c_j)^{x_i} P(c_j)}{\sum_{k=1}^{K} \prod_{i=1}^{n} P(t_i|c_k)^{x_i} P(c_k)}$$

c_j represents the document class. d represents the document. t_i represents word i. $P(c_j|d)$ represents the probability of document d being of class c_j.

$P(t_i|c_j)$ represents the probability of word t_i occurring in a document of class c_j. x_i represents the frequency of word t_i in the document.

n represents the total number of words. K represents the total number of classes which our case is 2.

This formula is used to determine the class probabilities by calculating the frequency of each word in the document and using these frequencies to make probability calculations, while ignoring dependencies between words and treating each word separately.

2.8. Logistic regression

According to KOLUKISA 2021, Logistic regression is a classification algorithm employed in machine learning. Its main objective is to estimate the probability of a binary outcome, such as a "yes" or "no" response, by considering

input variables. Logistic regression calculates the probability of an event by establishing the connection between a dependent variable and one or multiple independent variables. The logistic regression formula is utilized to model this relationship and make predictions.

$$p(y = 1|x) = 1/\left(1 + \exp\left(-(b0 + b1 * x1 + b2 * x2 + \cdots + bn * xn)\right)\right)$$

Here:

y represents the target variable or class. x represents the independent variables. $b0, b1, b2 \ldots bn$ represent the model parameters. The $\exp(bn * xn)$ function performs the mathematical operation e^x. In Logistic Regression, the correlation between the dependent and independent variables is computed to predict the probability of an event occurrence. This relationship is then utilized to make probabilistic predictions. This algorithm is commonly used in deep learning and was one of the classifiers analyzed in a study on Turkish character usage in text classification.

3. Results

Due to the increase in social media usage, the problem of cyberbullying has become more prominent, and there has been limited research on the issue in the Turkish language. While offensive language on social media is often defended as freedom of speech, this is not a tenable position. Moreover, manually filtering the growing amount of content on social media is becoming increasingly difficult, and developing automated solutions is becoming necessary. Additionally, it has been observed that Turkish uses suffixes rather than prefixes to create word meanings, which may suggest the importance of suffixes in Turkish compared to English.

We aimed to develop a text classification model to detect offensive language in Turkish text data, including posts from various forum sites, such as Twitter and Eksi Sozluk. We started by collecting a total of 11,253 posts and manually labeling them as offensive or non-offensive using a predefined list of offensive words. We then used index-based encoding and word2vec embedding to represent the text data numerically and fed the resulting features into several machine learning models, including LSTM, SVM, Random Forest Classifier, Extra Tree Classifier, Gaussian NB, Multinomial NB, and Logistic Regression. After evaluating the performance of these models. There are several ways to understand the performance of a model. These are confusion matrix, classification report, Receiver Operating Characteristic - ROC, accuracy.

3.1. Confusion matrix

A confusion matrix serves as an assessment tool for evaluating the effectiveness of a classification model. It accomplishes this by comparing the predicted classifications of the model with the actual classifications, enabling the determination of the number of correct and incorrect classifications. The confusion matrix is comprised of four fundamental terms, namely true positives, false positives, true negatives, and false negatives [18].

Table 1 illustrates an example of a confusion matrix, which is commonly used to evaluate the performance of a classification model. In this table, the rows represent the actual classifications, while the columns correspond to the model's predicted classifications. The terms True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) are organized within the matrix to demonstrate how the outcomes are categorized based on the model's predictions and the actual results. This structure provides valuable insights into the model's accuracy and areas where it might require improvement.

Table 1. Confusion matrix example

	Predicted Positive	Predicted Negative
True Positive	TP	FN
True Negative	FP	TN

Within this tabular representation, as outlined by Karimi (2021), the elements TP, FP, TN, and FN illustrate the correspondence between the predicted and actual classifications generated by the model. Specifically, TP denotes the count of true positive predictions, FP denotes the count of false positive predictions, TN denotes the count

of true negative predictions, and FN denotes the count of false negative predictions.

Sensitivity is the ratio of true positive (TP) examples to total positive (TP + FN) examples. Specificity is the ratio of true negative (TN) examples to total negative (TN + FP) examples. When the threshold changes, sensitivity and specificity rates change, and the ROC curve displays different values for these rates.

To gain insights into the performance of a classification model, the confusion matrix, as described by Karimi (2021), is employed to compute performance metrics including accuracy, precision, recall, and F1 score. These metrics serve the purpose of providing a comprehensive understanding of the classification model's performance.

3.2. Classification report

The classification report is a summary report that outlines the performance of a classification model. This report displays performance metrics such as accuracy, precision, recall, and F1 score for each class. These performance metrics are used to determine how well the model performed when classifying data [19].

In the classification report, accuracy represents the ratio of correctly classified examples by the model. Precision represents the ratio of correctly predicted positive examples for a given class. Recall represents the ratio of correctly predicted positive examples over all positive examples for a given class. F1 score represents the balance between precision and recall.

Macro average is the equal-weighted average of the performance metrics for each class. Weighted average is the weighted average of the performance metrics based on the proportion of class examples.

This classification report summarizes the performance of the model for each class and helps analyze the model's performance.

3.3. Receiver Operating Characteristic (ROC)

ROC is a curve and metric used to measure the performance of a classification model. The ROC curve visualizes the sensitivity and specificity rates provided by a model at different thresholds [14].

An ROC curve represents the performance of an ideal classification model, which is determined by how close the curve is to the top-left corner. A curve close to this area represents a model that provides high sensitivity and high specificity.
Additionally, the area under the ROC curve (AUC) is a metric used to measure the performance of the model. As the AUC value approaches 1, the model's performance is better. As the AUC value approaches 0.5, the model is randomly classifying examples.

3.4. Accuracy

Accuracy is a metric that shows the ratio of correctly classified examples by a classification model. The accuracy of a model is calculated by dividing the number of correctly classified examples by the total number of examples [20].

Accuracy rate is an important metric used to measure the performance of a model, but it is not sufficient on its own. Especially in imbalanced datasets, it can mislead the model's performance. For example, if there is a large imbalance between classes in a dataset and one class has much more examples, the model can give a high accuracy result without making correct classifications.
Therefore, other performance metrics should also be used in addition to accuracy. These metrics include confusion matrix, precision, recall, and F1 score, which indicate imbalances between classes and help to evaluate the performance of the model more accurately.

3.5. Model results

We implemented multiple algorithms to achieve the best results. Among the algorithms we tried are Gaussian Naive Bayes, Multinomial Naive Bayes, Logistic Regression, SVM, and LSTM. While obtaining these results, we tried to achieve

the best result by using different parameters for each algorithm.

When we look at the results, with word2Vec Gaussian Naïve bayes and SVM achive better performance. And for the random forest classifier, without word2vec gave us better result. Just after the results, a short summary will be shown in the table in the form of information.

3.5.1. Random forest classifier

When comparing the performance of different models, we evaluated their success based on the Area Under the Curve (AUC) score, a robust measure for assessing classification models. Among these, the random forest classifier stood out, delivering superior results. Specifically, we observed that employing a random forest classifier led to more accurate predictions and higher reliability compared to alternative models such as logistic regression, Gaussian Naive Bayes, SVM, and LSTM.

With an accuracy of 0.82, we surpassed the logistic regression, Gaussian Naive Bayes, SVM, and LSTM models, achieving the best accuracy. Figure 9 demonstrates the confusion matrix for the random forest classifier without the use of Word2Vec, providing insight into the distribution of true positives, false positives, true negatives, and false negatives. Additionally, Figure 10 illustrates the ROC curve for the random forest classifier without Word2Vec, showcasing its performance in terms of the true positive rate (sensitivity) and false positive rate.

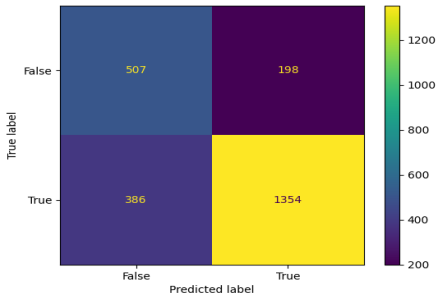

Figure 9. Confusion matrix for random forest classifier without word2vec

In the confusion matrix, there are TP = 507, FP = 198, FN = 386, and TN = 1354. Based on these values, the classifier correctly predicted the true positives (TP) and true negatives (TN). However, it also made some false positives (FP) and false negatives (FN), indicating that the classifier's performance is not ideal.

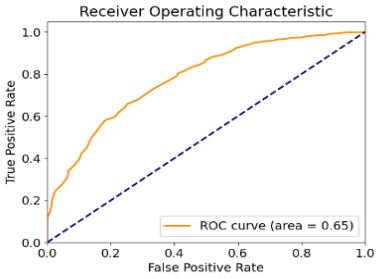

Figure 10. ROC Curve for Random Forest Classifier without Word2vec

Different metrics can be calculated based on this matrix to evaluate the performance of the classifier. For instance, metrics such as precision and recall can show how well the classifier detected true positives and false negatives. Other metrics such as F1-score balance precision and recall evaluating the overall performance of the classifier. The performance evaluation of a binary classifier involves utilizing the AUC (Area Under the Curve) score. This metric assesses the classifier's capability to differentiate between positive and negative classes. A score of 65 signifies that the classifier's performance surpasses that of random guessing.

When the worst result was considered, it was observed that it had an AUC score of 0.59. Regarding the accuracy value, a result of 0.58 was obtained. While it was initially assumed that utilizing Word2Vec would yield better results, it was observed that progressing through vectors led to improved outcomes in this case. Figure 11 presents the confusion matrix for the Random Forest Classifier with Word2Vec, showcasing the classification performance across true positives, false positives, true negatives, and false negatives. Meanwhile, Figure 12 illustrates the ROC curve for the Random Forest Classifier with Word2Vec, highlighting the trade-off between the true positive rate and false positive rate.

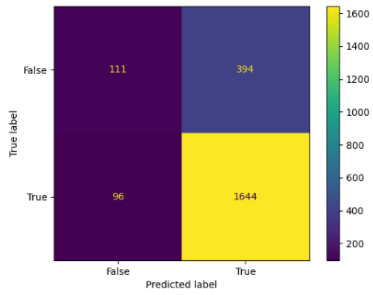

Figure 11. Confusion matrix for random forest classifier with Word2vec

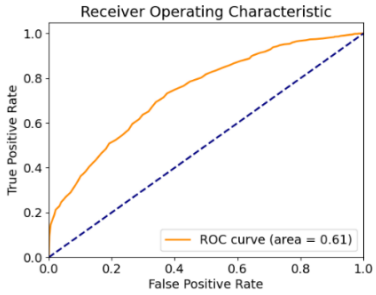

Figure 12. ROC Curve for random forest classifier with Word2vec

When we tested the random forest classifier model using word2vec, it was observed that there was low performance, although it did not differ much from its unused state, as seen in the figure. We can see in Figure 12 that the AUC score has decreased from 0.65 to 0.61.

3.5.2. Gaussian naïve bayes

If we examine a different algorithm, the Gaussian Naive Bayes, we observed that it achieved an accuracy value of 0.79. When we analyzed the AUC score, it resembled the performance of the Random Forest Classifier, yielding a value of 0.53. Notably, this result was obtained without utilizing Word2Vec in the Gaussian Naive Bayes model. Figure 13 illustrates the confusion matrix for Gaussian Naive Bayes without Word2Vec, detailing the distribution of true positives, false positives, true negatives, and false negatives. Additionally, Figure 14 depicts the ROC curve for Gaussian Naive Bayes without Word2Vec, visualizing the relationship between the true positive rate and false positive rate.

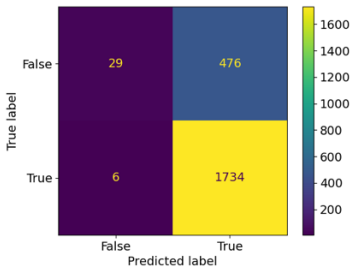

Figure 13. Confusion matrix for gaussian naive bayes without word2vec

Even when considering the highest performance achieved by the Gaussian Naive Bayes model, it fails to deliver the desired outcome or satisfactory results. However, in terms of accuracy, it achieved a respectable value of 0.79.

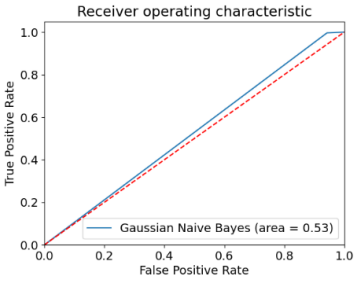

Figure 14. ROC Curve for gaussian naive bayes without word2vec

This result was obtained using Word2Vec as part of the feature representation. Figure 15 illustrates the confusion matrix for Gaussian Naive Bayes with Word2Vec, providing insights into the model's performance across true positives, false positives, true negatives, and false negatives. Moreover, Figure 16 shows the ROC curve for Gaussian Naive Bayes with Word2Vec, highlighting the balance between sensitivity and specificity.

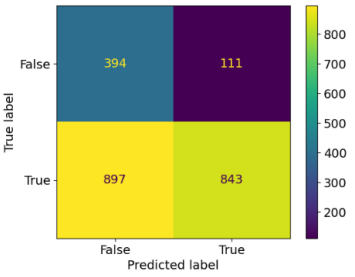

Figure 15. Confusion matrix for gaussian naive bayes with word2vec

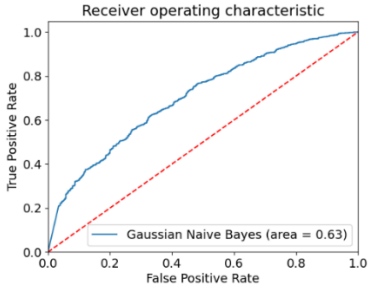

Figure 16. ROC for gaussian naive bayes with word2vec

If we use Word2Vec, it is observed that the AUC score improves significantly. However, the accuracy value drops to 0.55, indicating that the model struggles with correct classifications. Upon examining the confusion matrix in Figure 15, it becomes evident that the model produces a high number of false positives, which affects its overall performance and reliability.

3.5.3. Logistic regression

If we look at another model, namely logistic regression, it is observed that the accuracy value reaches 0.79 when Word2Vec is not used. However, this accuracy drops slightly to 0.77 when Word2Vec is applied. The AUC score and the corresponding confusion matrix for logistic regression without Word2Vec are depicted in Figure 17 and Figure 18, respectively.

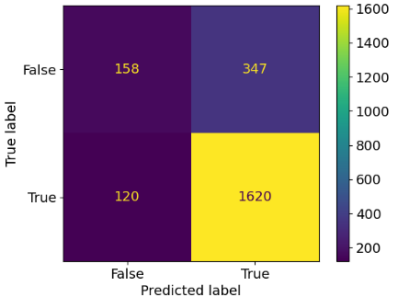

Figure 17. Confusion matrix for logistic regression without Word2vec

Figure 18. ROC for logistic regression without word2vec

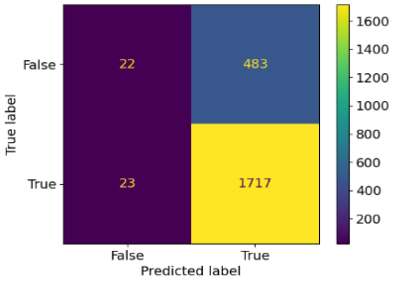

Figure 19. Confusion matrix for logistic regression with word2vec

Although the accuracy value has not changed significantly, it is observed that the number of lines predicted by the model has decreased. Similarly, when examining the ROC curve, a noticeable drop in the score is observed. Specifically, the value decreased from 0.62 to 0.52.

Figure 20. ROC for logistic regression with word2vec

While the use of Word2Vec can be important in certain scenarios, the tests conducted with logistic regression reveal that its impact is not particularly significant. In fact, in the worst-case scenario, the use of Word2Vec leads to a decline in accuracy. These observations are illustrated in Figure 19 and Figure 20, which show the confusion matrix and ROC curve for logistic regression with Word2Vec, respectively.

4.5.4. Support Vector Machine (SVM)

When examining the Support Vector Machine (SVM) algorithm, it is observed that there is an increase in the AUC score when Word2Vec is not used. Specifically, the AUC score, which was 0.62 without Word2Vec, decreased to 0.50 after incorporating Word2Vec. These results suggest that using Word2Vec negatively impacted the model's performance in this case. The confusion matrix and ROC curve for SVM without Word2Vec are shown in Figure 21 and Figure 22, respectively.

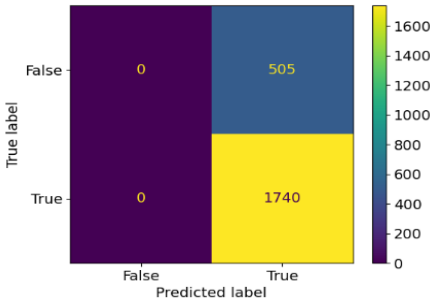

Figure 21. Confusion matrix for SVM without word2vec

Despite the seemingly high AUC score of the model, a closer examination of the confusion matrix reveals that the incorrect predictions are not truly incorrect.

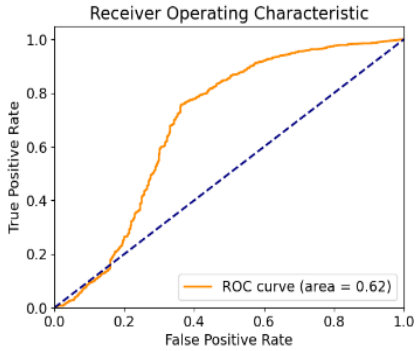

Figure 22. ROC for SVM without word2vec

However, the accuracy value is observed to be 0.77 when Word2Vec is not used. This result is depicted in Figure 23 and Figure 24.

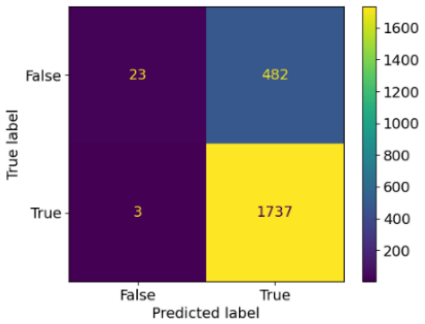

Figure 23. Confusion matrix for SVM with word2vec

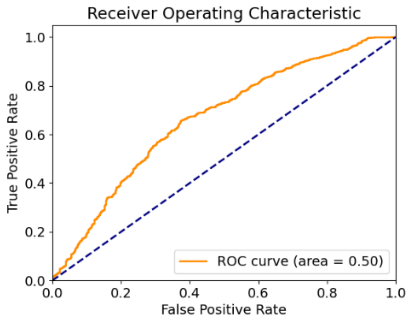

Figure 24. ROC for SVM with word2vec

When Word2Vec is utilized, the AUC score decreases, although the accuracy value remains the same as before. Upon examining the confusion matrix, it becomes evident that while there are a few misclassified instances, the overall number of misclassifications is minimal.

3.5.5. Long Short-Term Memory (LSTM)

When evaluating the performance of LSTM, a leading algorithm in artificial intelligence, its AUC score stands out, showcasing its ability to capture long-term dependencies and complex patterns in sequential data. However, despite this strength, LSTM's accuracy does not surpass that of the Random Forest model, indicating that while LSTM excels in probabilistic differentiation, it may be less consistent in making precise classifications.

As shown in Figure 25, the confusion matrix for the Support Vector Machine (SVM) without using Word2Vec embeddings provides a good classification result when compared to other models.

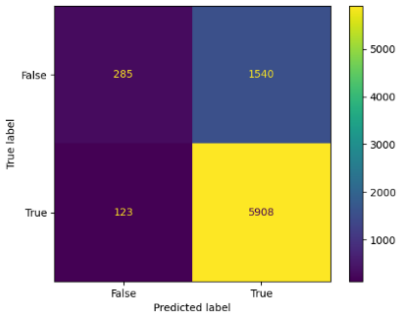

Figure 25. Confusion matrix for SVM without word2vec

Similarly, Figure 26 presents the ROC curve for SVM without Word2Vec, illustrating the model's performance.

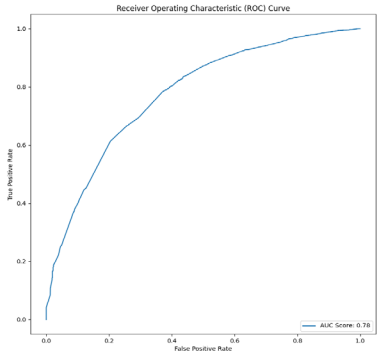

Figure 26. ROC for SVM without word2vec

On the LSTM model, when using only vectors without Word2Vec, the results are impressive, with an accuracy value of 0.78 and an AUC score of 0.79. This result is quite close to the performance of Random Forest. However, a closer look at the confusion matrix (as shown in Figure 25) reveals that while the results are similar, Random Forest achieves a slightly higher accuracy. On the other hand, when comparing AUC scores, Figure 26 clearly demonstrates that LSTM outperforms Random Forest, making it a better choice for distinguishing between classes.

4. Conclusion

In conclusion, this dissertation has conducted an extensive investigation into the identification and analysis of offensive and discriminatory language within the specific context of the Turkish language. Throughout this scholarly inquiry, the dissertation has emphasized the fundamental role of language in facilitating effective communication and its profound impact on human development. It has also underscored the urgent need to address the detrimental consequences of hate speech on individuals and society as a whole.

The study commenced by acknowledging the historical significance of natural language processing (NLP) during World War II, which laid the groundwork for subsequent advancements in this transformative technology. Furthermore, it has shed light on the limited availability of data for studying hate speech in the Turkish language compared to the extensive research conducted in English. This scarcity of data has highlighted the critical importance of generating relevant and contextually appropriate data within the Turkish linguistic domain, particularly in order to understand and mitigate the psychological impacts of inappropriate language use, especially among vulnerable populations such as children.

Moreover, the dissertation has argued that hate speech goes beyond mere violations of freedom of expression, as it has the potential to reinforce biases, perpetuate discrimination, and incite acts of violence. Despite the implementation of community standards and reporting mechanisms on digital platforms, hate speech continues to prevail, necessitating additional measures to ensure the safety and well-being of internet users. To address these challenges, the dissertation has proposed several valuable contributions, including the systematic collection of offensive language data from online sources, the utilization of preprocessing techniques such as stemming, suffix analysis, and censorship to optimize the input data, and the development and evaluation of deep learning models using comprehensive datasets from popular internet platforms. Various classification algorithms have been employed, and the results have been assessed using key performance metrics, such as accuracy, F-1 score, and confusion matrix, with a specific focus on offensive text in the Turkish language.

Through this research, significant insights and tools have been developed to enhance Turkish language processing and facilitate the creation of applications that prioritize child safety. Moreover, this investigation contributes to broader efforts aimed at combating prejudice, discrimination, and intolerance by enabling the detection and analysis of cyberbullying within the Turkish linguistic realm. Ultimately, this dissertation underscores the importance of addressing hate speech, particularly in the Turkish language, and establishes a solid foundation for future research endeavors and practical applications in the field of natural language processing. The ultimate goal remains the protection of individuals, the promotion of tolerance, and the establishment of a secure online environment for all users.

Article Information Form

Acknowledgments
Authors would like to thank Dr. Rüştü Murat Demirer for his contributions.

Funding
Authors has no received any financial support for the research, authorship or publication of this study.

Authors' Contribution
Authors contributed equally to the study.

The Declaration of Conflict of Interest/ Common Interest
No conflict of interest or common interest has been declared by authors.

The Declaration of Ethics Committee Approval
This study does not require ethics committee permission or any special permission.

The Declaration of Research and Publication Ethics
Authors of the paper declare that they comply with the scientific, ethical and quotation rules of SAUJS in all processes of the paper and that they do not make any falsification on the data collected. In addition, they declare that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science.

References

[1] E. Adalı, "Natural Language Processing," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 24, no. 2, pp. 1–17, 2016.

[2] S. Rosenthal, P. Atanasova, G. Karadzhov, M. Zampieri and P. Nakov, "SOLID: A Large-Scale Semi-Supervised Dataset for Offensive Language Identification," arXiv preprint arXiv:2004.14454, 2020.

[3] Ç. Çöltekin, "A Corpus of Turkish Offensive Language on Social Media," Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC 2020), pp. 1–8, 2020.

[4] C. Casula, A. P. Aprosio, S. Menini and S. Tonelli, "FBK-DH at SemEval-2020 Task 12: Using Multi-channel BERT for Multilingual Offensive Language Detection," Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval 2020), pp. 1–10, 2020.

[5] Ö. Anil and R. Yeniterzi, "SU-NLP at SemEval-2020 Task 12: Offensive Language Identification in Turkish Tweets," Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval 2020), pp. 1–8, 2020.

[6] L. Ma, Y. Liu, X. Zhang, Y. Ye, Yin and B. F. G. Johnson, "Deep Learning in Remote Sensing Applications: A Meta-analysis and Review," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 152, pp. 166–177, 2019.

[7] T. Mikolov, K. Chen, G. S. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," arXiv preprint arXiv:1301.3781, 2013.

[8] K. Potdar, T. S. Pardawala and C. D. Pai, "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," International Journal of Computer Applications, vol. 175, no. 4, pp. 7–9, 2017.

[9] W. Gao and Z. Zhou, "Towards Convergence Rate Analysis of Random Forests for Classification," Artificial Intelligence, vol. 313, p. 103788, 2020.

[10] O. C. Njoku, "Decision Trees and Their Application for Classification and Regression Problems," M.S. thesis,

Missouri State University, 2020. [Online]. Available: https://bearworks.missouristate.edu/theses/3406.

[11] E. K. Ampomah, Z. Qin and G. Nyame, "Evaluation of Tree-Based Ensemble Machine Learning Models in Predicting Stock Price Direction of Movement," Information, vol. 11, no. 6, p. 332, 2020.

[12] X. Lin, "Sentiment Analysis of E-commerce Customer Reviews Based on Natural Language Processing," Proceedings of the 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE), pp. 1–5, 2020.

[13] V. Apostolidis-Afentoulis, "SVM Classification with Linear and RBF Kernels," ResearchGate, 2015.

[14] J. I. Razin, A. Karim, M. F. Mridha, S. M. R. Rifat and T. Alam, "A Long Short-Term Memory (LSTM) Model for Business Sentiment Analysis Based on Recurrent Neural Network," in Lecture Notes on Data Engineering and Communications Technologies, Springer, pp. 1–15, 2021.

[15] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM: A Tutorial into Long Short-Term Memory Recurrent Neural Networks," ResearchGate, 2019.

[16] C. Naulak, "A Comparative Study of Naive Bayes Classifiers with Improved Technique on Text Classification," TechRxiv, 2022.

[17] A. A. Kolukisa, "Turkish Character Usage in Text Classification (JAIDA)," ResearchGate, 2021.

[18] Z. Karimi, "Confusion Matrix," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/355096788_Confusion_Matrix.

[19] S. Akram, "CLASSIFICATION REPORT," ResearchGate, 2021. [Online]. Available: https://www.researchgate.net/publication/357974052_CLASSIFICATION_REPORT.

[20] Q. Kong, W. Wang, D. Zhang and W. Zhang, "Two Kinds of Average Approximation Accuracy," CAAI Transactions on Intelligence Technology, 2023.