A Comparative Study of Word Embedding Techniques for Turkish Sentiment Analysis

Buse Sarıçayır[1][0009-0005-5868-0189], Caner Özcan [0000-0002-2854-4005], and Yasin Ortakçı [0000-0002-0683-2049]

[1] Karabuk University, Karabuk 78000, Türkiye
busesaricayir@gmail.com

Abstract. This paper presents a comparative analysis of three different word embedding techniques—TF-IDF, Word2Vec, and FastText in the context of Turkish sentiment analysis. Each method offers a unique trade-off between computational efficiency, representational richness, and classification accuracy of Turkish text. TF-IDF, a frequency-based approach, is computationally inexpensive but neglects word order and contextual information. Word2Vec generates context-aware embeddings capturing semantic relationships but remains context-independent for individual words. FastText further refines word representation by incorporating subword information, which is particularly advantageous for morphologically rich languages like Turkish. The performance of these embedding methods was evaluated using four different classification models (Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine) on the Winvoker Turkish sentiment analysis dataset. Our experimental results show that Word2Vec achieves the highest accuracy with Logistic Regression. These results contribute to a better understanding of the strengths and limitations of various word embedding techniques for sentiment analysis in morphologically rich languages.

Keywords: Sentiment Analysis, Turkish text, TF-IDF, Word2Vec, FastText, Embedding Models.

1 Introduction

Sentiment analysis, the computational process of identifying and categorizing opinions expressed in text, has become an indispensable tool for understanding human behavior and extracting valuable insights from the ever-increasing volume of unstructured data generated daily. Its applications are widespread, ranging from social media monitoring and brand reputation management to customer feedback analysis, market research, and political science [1]. The accuracy and effectiveness of sentiment analysis systems depend heavily on the ability to effectively represent the meaning and context of words within sentences and documents. This representation significantly affects the classifier's ability to detect subtle nuances in sentiment expressions, such as sarcasm, irony, and negation [2].

This study focuses on sentiment analysis of Turkish text, a particularly challenging domain due to the complexities of the Turkish language, including its agglutinative nature and rich morphology. The agglutinative structure, where multiple suffixes are

attached to a root word, requires sophisticated linguistic processing to accurately capture the intended meaning and sentiment. Furthermore, the prevalence of informal language and colloquialisms in online Turkish text presents additional obstacles for accurate sentiment classification. Addressing these challenges is crucial because effective sentiment analysis of Turkish text offers significant opportunities, including deeper understanding of public opinion in Turkey, improved market research for Turkish businesses, and enhanced customer service experiences for Turkish-speaking consumers. This study aims to contribute to the advancement of sentiment analysis techniques specifically tailored to the nuances of the Turkish language.

We conduct a comparative analysis of three word embedding methods—TF-IDF, Word2Vec, and FastText—for the task of Turkish sentiment analysis. TF-IDF, known for its computational efficiency and ease of implementation, represents words as vectors based solely on their frequency within a document and across the corpus [3]. However, it lacks the ability to capture contextual information and word order, both of which are critical for accurately conveying sentiment [4]. Word2Vec [5], on the other hand, generates context-aware embeddings that capture semantic relationships between words through their co-occurrence patterns [6]. Despite this advantage, Word2Vec treats each word independently of its context, which limits its ability to distinguish between multiple meanings of a word [7]. FastText addresses this limitation by incorporating subword information, such as n-grams, which is particularly advantageous for handling the morphological complexity of the Turkish language and alleviating challenges associated with rare or out-of-vocabulary words [8]. By providing a more nuanced representation of sentiment, FastText offers a potential improvement over the other methods. The analysis seeks to identify the most effective embedding technique for Turkish sentiment analysis [9].

This paper presents an analysis of three word embedding techniques for Turkish sentiment analysis, using the Winvoker dataset [10] as our corpus. Our experiments, conducted using the Winvoker Turkish sentiment dataset and four machine learning models, demonstrate that Word2Vec consistently outperforms TF-IDF and achieves accuracy comparable to FastText. These results highlight the importance of considering contextual information and subword morphology for effective sentiment analysis in morphologically rich languages like Turkish. A detailed analysis of the results, including confusion matrices and performance metrics for each model and embedding technique, is presented in the following sections.

2 Related Work

Recent research on word embedding methods for Turkish sentiment analysis reveals a trend towards increasingly sophisticated deep learning approaches, although simpler methods still have value in certain contexts. Early work relied on traditional machine learning techniques like Support Vector Machines (SVM) and Logistic Regression (LR). Demircan et al. [11], using Turkish e-commerce reviews, achieved an accuracy of 87% and an F1-score of 0.86 using SVM and feature selection, highlighting the importance of careful feature engineering even with simpler models. Similarly, Aydın

and Gungor [12] found that supervised techniques such as SVM and Random Forest (RF) outperformed semi-supervised and unsupervised methods (88% accuracy versus 70% and 65%, respectively) on various datasets including reviews and news articles. Kemaloglu et al. [13], working with noisy social media data, obtained an accuracy of 84% and F1-score of 0.82 using logistic regression with TF-IDF, showcasing the challenges inherent in such datasets. Ozdemir and Ortakci [14] further demonstrate the continued relevance of frequency-based methods by achieving 84% accuracy using TF-IDF on a call center dataset.

The introduction of transformer-based models has significantly improved performance. Acikalin et al. [15] achieved 90% accuracy and 0.89 F1-score using BERT on a custom Turkish sentiment dataset, outperforming traditional methods by 15%. Guven [16], using BERTurk with a novel text filtering technique on benchmark datasets, further enhanced accuracy to 92.5% and F1-score to 0.93, underscoring the impact of data preprocessing. Ba Alawi & Bozkurt [17] corroborated these findings, demonstrating that BERTurk outperformed traditional word embeddings by 10% in accuracy (91%) and F1-score (0.90) across diverse datasets. Alshammari & Akyüz [18] used a RoBERTa model fine-tuned for Turkish tweets to achieve an impressive F1-score of 0.92 in analyzing mental health-related sentiments. However, Cam et al. [19] showed that with careful feature engineering, Gradient Boosting could still achieve high accuracy (89%) and F1-score (0.88) on financial sentiment tweets.

Automation of model training is also promising. Tasar et al. [20] utilized AutoTrain to achieve 86% accuracy and 0.84 F1-score on various Turkish sentiment datasets, suggesting a streamlined approach for achieving near state-of-the-art results. Finally, Ozdemir and Ortakci [21] combined hybrid vectorization with deep learning models for a call center routing task, obtaining 85% accuracy and an F1-score of 0.83, highlighting the potential of tailored approaches for specific applications. Overall, these studies indicate that while sophisticated deep learning models currently dominate, simpler methods combined with effective feature engineering can still be competitive, especially in specialized domains with limited data or specific requirements.

While transformer-based models such as BERTurk and mBERT have demonstrated superior performance in Turkish sentiment analysis, this study intentionally focuses on classical embedding methods to provide a controlled comparison of TF-IDF, Word2Vec, and FastText. Transformer-based approaches are left for future work.

3 Methodology

This study employs a comparative analysis of three different word embedding techniques for Turkish sentiment analysis. The methodology begins with preprocessing of the Winvoker Turkish sentiment analysis dataset, cleaning and preparing the text data for vectorization. Each word embedding technique generates a distinct vector representation of the words in the dataset. These vector representations are then used as input features for four different machine learning classification models: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). The performance of each model, combined with each embedding tech-

nique, is evaluated using accuracy, precision, recall, and F1-score metrics. This comparative analysis allows for a detailed assessment of the strengths and weaknesses of each word embedding method in the context of Turkish sentiment analysis and determines the most effective combination for achieving optimal classification performance.

3.1 Data Preprocessing

The preprocessing pipeline implemented in this study included a series of carefully designed transformations to prepare the textual data for subsequent analysis. The raw text data went through several basic procedures: Firstly, a filtering process was applied to retain only the characters of the Turkish alphabet (both uppercase and lowercase letters and additional characters specific to the Turkish orthography) and whitespace characters. All other characters, such as punctuation marks, numbers, and special symbols, were removed to reduce dimensionality and noise in the data. This decision simplifies the feature space while acknowledging the possibility of losing semantically relevant information conveyed through punctuation marks or other non-alphabetic characters.

The entire text was then converted to lowercase to standardize the word representation and eliminate case sensitivity as a differentiating factor in the analysis. This step ensured consistency and prevented the algorithm from treating changes in case usage as separate word units. Finally, leading and trailing spaces were removed from each text string to create a clean and consistent format for subsequent processing.

After text cleaning, categorical sentiment labels were converted to numerical representations using a label encoding scheme. This transformation mapped the original sentiment categories (e.g., positive, negative, neutral) to numerical values and ensured compliance with the input requirements of the machine learning models used in this study. The specific mapping used was carefully recorded and reported to ensure full transparency and ease of interpretation of the results. This coding step ensured that the classification models received numerical input for the target variable, a necessary condition for many machine learning algorithms. Morphological preprocessing steps such as stemming and lemmatization were not applied in this study, as the focus was on evaluating the embedding methods under a standardized preprocessing pipeline. The impact of such techniques on model performance remains an avenue for future research.

3.2 Text Vectorization

In this study, TF-IDF, Word2Vec and FastText word embedding models are compared. Based on this comparison, it is determined which embedding model is better for the dataset used.

TF-IDF. TF-IDF, a fundamental technique in information retrieval and text mining, represents words as vectors based on their frequency within a document and their inverse frequency across the entire corpus [3]. While simple and computationally

inexpensive, TF-IDF suffers from a significant limitation: it ignores word order and contextual information, leading to a potentially impoverished representation of sentiment. This can be particularly problematic when dealing with complex sentences, where the sentiment expressed is highly dependent on word order and the relationships between words [4].

TF-IDF takes into account both the frequency of a word in a document and the prevalence of that word in the corpus. This allows it to identify terms that are more important to the document than they would be if considered alone [22]. By identifying the most important terms, TF-IDF is a useful tool for analyzing and understanding text. This model helps to identify and prioritize the most relevant words in a text [23]. It combines two key metrics: term frequency (TF) and inverse document frequency (IDF). TF measures the frequency of a specific term within a document, normalized by the total number of terms in that document:

$$TF = \frac{Number\ of\ times\ the\ term\ appears\ in\ the\ document}{Total\ number\ of\ terms\ in\ document} \quad (1)$$

A higher TF value indicates a greater importance of the term within the specific document. IDF, conversely, measures the inverse frequency of a term across the entire corpus. It's calculated as:

$$IDF(t) = log(\frac{N}{df(t)}) \quad (2)$$

where N is the total number of documents in the corpus, and $df(t)$ is the number of documents containing term t. A higher IDF value signifies that the term is rare across the corpus, indicating greater discriminative power. The logarithm is used to dampen the effect of very high document counts. TF-IDF combines these two metrics to generate a weighted score for each term in each document:

$$TF\text{-}IDF = TF * IDF \quad (3)$$

This score measures the significance of a term in a document based on both its frequency within that document and its relative rarity across the overall corpus. Terms that appear frequently within a particular document but are rare across the corpus will receive a high TF-IDF score, reflecting their importance in characterizing the content of that document [24]. Conversely, terms that are frequent across the entire corpus will have a lower IDF score, reducing their overall TF-IDF weight. This weighting scheme allows TF-IDF to effectively highlight terms that are both frequent within a specific document and distinctive across the broader collection, making it a powerful technique for feature extraction in text analysis tasks, such as sentiment classification [25].

Word2Vec. Word2Vec addresses this limitation by generating dense vector representations (embeddings) of words that capture semantic relationships between words based on their co-occurrence in a large corpus of text [5]. Word2Vec models, such as

Continuous Bag-of-Words (CBOW) and Skip-gram, learn contextual relationships, leading to more nuanced representations than TF-IDF [26]. However, Word2Vec embeddings are still context-independent, meaning that the same word has the same vector representation regardless of its usage within a sentence [7].

Word2Vec represents a family of model architectures designed to generate distributed word embeddings that map words to dense vectors of real numbers within a continuous vector space [27]. The underlying principle is that semantically similar words will exhibit proximity within this space. This is achieved through the training of shallow, two-layer neural networks on extensive textual corpora. Two primary architectures are commonly employed: CBOW and Skip-gram. CBOW predicts a target word based on its surrounding context words, while Skip-gram performs the inverse operation, predicting context words given a target word [28]. Both approaches are based on the distributional hypothesis, which states that words with similar contexts tend to have related meanings.

While a singular, concise formula cannot fully encapsulate the complexities of Word2Vec, the Skip-gram architecture can be elucidated conceptually. The objective function of Skip-gram aims to maximize the probability of observing context words given a central word.

w(i): The central word within a given window of context
c(i): A context word surrounding w(i)
v(w(i)): The input vector representation of w(i)
u(c(i)): The output vector representation of c(i) (4)

The Skip-gram model seeks to maximize the following likelihood function across the entire corpus:

$$\textit{Maximize: } \prod \Big(\big(w(i), c(i)\big) \in Corpus \Big) P(c(i)|w(i)) \tag{5}$$

The conditional probability P(c(i) | w(i)) is often modeled using a SoftMax function:

$$P(c(i)|w(i)) = \frac{exp\ (u(c(i)) \cdot v(w(i)))}{\sum (j \in Vocabulary) exp\ (u(j) \cdot v(w(i)))} \tag{6}$$

where '·' denotes the dot product. Model training involves iterative adjustment of the input and output vectors (v and u) to optimize this likelihood function, typically employing stochastic gradient descent or similar optimization algorithms. The resulting word embeddings are commonly represented by the learned input vectors (v), encapsulating the semantic relationships derived during the training process. It is important to note that this description represents a simplified representation; practical implementations often incorporate efficiency-enhancing techniques such as negative sampling or hierarchical SoftMax to address the computational challenges associated with the SoftMax normalization over the entire vocabulary. Nevertheless, this formulation provides a fundamental understanding of the mathematical underpinnings of the Skip-gram model within the Word2Vec framework.

FastText. FastText, building upon the strengths of Word2Vec, further refines the representation of words by incorporating subword information [29]. Unlike Word2Vec, which treats each word as an atomic unit, FastText considers the internal structure of words by generating embeddings not only for the entire word but also for its constituent n-grams (subsequences of characters). This innovative approach proves particularly advantageous when dealing with morphologically rich languages like Turkish, where a single word can encompass multiple morphemes carrying distinct semantic contributions [8]. The inclusion of n-gram embeddings allows FastText to effectively represent rare words, out-of-vocabulary terms (OOV), and morphologically complex words that might be inadequately captured by Word2Vec [30]. This enhanced representation provides a more robust and nuanced understanding of word meaning, leading to improved performance in downstream tasks such as sentiment analysis, especially when faced with the challenges presented by the complexities of Turkish grammar and vocabulary. By capturing subword information, FastText effectively bridges the gap between the limited context awareness of TF-IDF and the context-independent nature of standard Word2Vec, providing a powerful approach for creating rich and informative word embeddings [9].

Unlike Word2Vec, which assigns a single vector per word, FastText generates embeddings for both the entire word and its constituent n-grams [29, 31]. This allows FastText to handle rare words and OOV terms more effectively, a particularly valuable feature for morphologically rich languages.

While, like Word2Vec, a single formula doesn't fully capture FastText's complexity, we can outline a general framework for its objective function. Similar to Word2Vec, the goal is to learn vector representations that maximize the likelihood of observing context words given a target word.

$$\begin{gathered} V: \text{The vocabulary of words} \\ w(i): \text{The central word within a given window of context} \\ c(i): \text{A context word surrounding } w(i) \\ G(w(i)): \text{The set of n grams that constitute word } w(i). \\ \text{This includes the word itself as a special n-gram.} \\ v(g) \in R(d): \text{The d dimensional vector embedding for n gram } g. \\ \Theta: \text{The set of all model parameters} \end{gathered} \tag{7}$$

The objective function for FastText aims to maximize the log-likelihood of the observed word-context pairs in the training corpus:

$$\text{Maximize: } L(\Theta): \sum((w(i), C(i)) \in \text{Corpus}) \log P(C(i)|w(i); \Theta) \tag{8}$$

Where:

- C(i): Represents the context words associated with w(i)
- P(C(i) | w(i); Θ): The probability of observing the context words C(i) given word w(i), parameterized by Θ. This probability is calculated considering the embeddings of n grams composing w(i).

The crucial difference from Word2Vec lies in the calculation of P(C(i) | w(i); θ). Instead of relying solely on the embedding of w(i), FastText uses a combination of the embeddings of its constituent n-grams:

$$\textit{Representation of } w(i) = \sum(g \in G(w(i)))v(g) \tag{9}$$

This aggregated representation is then used in the probability calculation, typically using a SoftMax function (or its approximations, such as negative sampling) similar to Word2Vec. The model parameters θ are learned through optimization techniques like stochastic gradient descent to maximize the log-likelihood. The learned n-gram embeddings v(g) are then used to represent words and subwords, offering a more nuanced and robust representation compared to Word2Vec, particularly beneficial for morphologically rich languages and handling OOV words. The specific implementation details, such as the choice of n-gram lengths and the optimization algorithm, can affect the performance of the model.

3.3 Machine Learning Classifiers

Logistic Regression (LR). LR [32] is a statistical method used for binary and multi-class classification tasks. It models the relationship between one or more independent variables and a categorical dependent variable using a logistic function. The decision boundary is determined by a linear combination of the input features and their corresponding weights, learned during training. LR is computationally efficient, interpretable, and effective when the data exhibits a linear separable pattern. However, its performance can degrade in cases of multicollinearity or non-linearity without appropriate preprocessing or feature engineering.

Decision Tree (DT). A DT [33] is a non-parametric, tree-structured algorithm used for both classification and regression tasks. It divides the feature space into regions based on recursive binary splits guided by a criterion such as Gini impurity or information gain. Each internal node represents a decision based on a single feature, while the leaf nodes correspond to the predicted results. DTs are intuitive, easy to visualize, and can handle non-linear relationships effectively. However, they are prone to overfitting, especially when the tree depth is not restricted, and their performance can be sensitive to small variations in the data.

Random Forest (RF). RF [34] is an ensemble learning method that builds multiple decision trees during training and aggregates their predictions to improve classification or regression performance. Each tree is constructed using a random subset of the training data and a random selection of features, which introduces diversity and reduces overfitting. The final prediction is typically determined through majority voting for classification or averaging for regression tasks. RF models are robust to noise, can handle high-dimensional data, and are relatively immune to overfitting. However, they can be computationally intensive for large datasets and lack interpretability compared to simpler models.

Support Vector Machine (SVM). SVM [35] is a supervised learning algorithm that aims to find the optimal hyperplane to separate data points belonging to different classes with the maximum margin. It uses kernel functions, such as linear, polynomial, or radial basis functions, to handle non-linear separability by transforming the input space into a higher-dimensional feature space. SVMs are particularly effective for datasets with clear margins of separation and are less susceptible to overfitting in high-dimensional spaces. However, they require careful parameter tuning and can be computationally expensive for large datasets.

4 Experimental Results

The optimum parameters of the classifiers used in the experiments were carefully determined to maximize the performance of each model and are presented in Table 1. In this parameter optimization process, hyperparameters for each classifier were examined with a comprehensive screening method and the combinations that provided the best results were selected. This process is a critical step to increase model accuracy and ensure the reliability of the experimental results, while also ensuring that the comparison between different classifiers is fair. The parameter values listed in Table 1 represent the optimized results according to the structures of the relevant algorithms and the dataset.

Table 1. Optimal Parameters for Classifiers

Model	Parameter	Value
TF-IDF	max_features	10000
	ngram_range	(1, 2)
	stop_words	None
Word2Vec	vector_size	300
	window	5
	min_count	1
	workers	4
FastText	vector_size	100
	window	5
	min_count	5
	workers	3
	word_ngrams	1
LR	max_iter	1000
RF	n_estimators	10
SVM	max_iter	1000

The analyses we conducted in this study were performed using the Winvoker dataset. Winvoker's training dataset consists of 489,644 instances of 262,166 positive, 56,561 negative and 170,917 neutral classes, compiled from different sources such as humir,

e-commerce site comments and Wikipedia. There are a total of 48965 examples in the test data set. The class distribution in the test data set is as shown in Fig 1. Most sentiment models have only two labels; positive and negative. However, the user input can be entirely notr sentences. Positive and negative sentences were taken from comments, while notr sentences were extracted from the Turkish wiki transcript. Table 2 shows an example of each class.

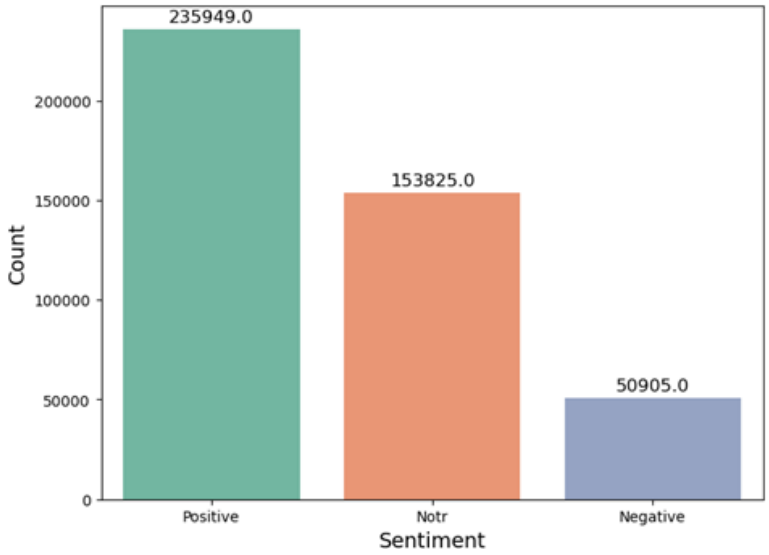

Fig. 1. Class distribution in test dataset.

Table 2. Data examples of classes from the dataset.

Text	Label	Dataset
hızlı kargo, temiz alışveriş. teşekkür ederim.	Positive	urun_yorumlari
Çünkü aranan tapınak bu bölgededir.	Notr	wiki
özellikle ışığı çok yetersiz ve 4 tane değil 2 tane geliyor bana siyah geldi ancak ışığı dediğim gibi yetersiz	Negative	urun_yorumlari

The performance of the proposed models was evaluated using a set of standard classification metrics, ensuring a comprehensive assessment. Accuracy (10) was employed as a primary metric, representing the proportion of correctly classified instances to the total number of instances, providing an overall measure of the model's predictive capability. Precision (11), defined as the proportion of true positive predictions out of all positive predictions, was used to evaluate the model's ability to minimize false positive rates, which is critical in scenarios where false alarms carry significant consequences.

In addition to precision, recall (12) was utilized to evaluate the model's efficacy in identifying actual positive cases, a particularly salient aspect in tasks were failing to detect positives can be costly. The F1-score (13), representing a harmonic mean of precision and recall, was incorporated to balance the trade-off between these two metrics, a strategy that is especially valuable for imbalanced datasets. The confusion matrix, a tool that provides a detailed breakdown of the model's predictions, was also employed to illuminate the model's strengths and weaknesses. This comprehensive analysis, facilitated by the incorporation of various metrics, ensured a robust and multi-faceted evaluation of model performance.

$$Accuracy = (TP+TN)/(TP+TN+FN+FP) \quad (10)$$

$$Precision = TP/(TP+FP) \quad (11)$$

$$Recall = TP/(TP+FN) \quad (12)$$

$$F1{-}Score = 2*((Precision*Recall)/(Precision+Recall)) \quad (13)$$

4.1 Results

To compare the performance of the three word embedding techniques (TF-IDF, Word2Vec, and FastText) in the context of Turkish sentiment analysis, we trained four different machine learning models: LR, DT, RF, and SVM. Each embedding technique was used with each model, resulting in twelve separate model configurations. The following metrics were calculated for each configuration: Accuracy, precision, recall, f1-score, and time. The results are summarized in Table 3.

Table 3. Performance of Sentiment Analysis Models Across Embedding Techniques

Embedding	Model	Accuracy	Precision	Recall	F1-Score	Time
TF-IDF	LR	0.633	0.60	0.63	0.61	59.74
TF-IDF	DT	0.585	0.58	0.59	0.58	410.19
TF-IDF	RF	0.589	0.58	0.59	0.58	199.50
TF-IDF	SVM	0.376	0.54	0.38	0.28	73.60
Word2Vec	**LR**	**0.911**	**0.91**	**0.91**	**0.91**	**390.08**
Word2Vec	DT	0.845	0.85	0.85	0.85	412.84
Word2Vec	RF	0.895	0.89	0.90	0.89	97.09
Word2Vec	SVM	0.331	0.31	0.33	0.32	443.42
FastText	LR	0.767	0.75	0.77	0.75	225.36
FastText	DT	0.683	0.69	0.68	0.68	265.06
FastText	RF	0.760	0.74	0.76	0.75	89.59
FastText	SVM	0.554	0.59	0.55	0.48	440.97

The results show a clear performance advantage for Word2Vec in at least three of the four machine learning models. In particular, LR yielded the highest accuracy scores across all three embedding methods. With Word2Vec, LR reached an accuracy of 0.911, significantly outperforming other models using Word2Vec.

The results clearly demonstrate the superiority of Word2Vec and, to a lesser extent, FastText over TF-IDF. TF-IDF, a simple bag-of-words approach, relies solely on word frequencies and ignores word order and semantic relationships. This limitation is reflected in its significantly lower accuracy scores across all models compared to the other two embedding methods. The average accuracy for TF-IDF is approximately 0.54, while Word2Vec achieves an average accuracy of roughly 0.79 and FastText an average of 0.70. Word2Vec and FastText, on the other hand, generate dense vector representations of words that capture semantic similarities. This leads to better performance in downstream tasks like text classification. Word2Vec consistently outperforms FastText, suggesting that its approach to generating word embeddings is more effective for this specific dataset and task. However, the training time for Word2Vec is noticeably longer, particularly for LR and DT models. FastText provides a good balance between accuracy and speed, making it a potentially attractive option when computational resources are limited.

Comparison of the machine learning models reveals distinct performance characteristics. LR achieves remarkable accuracy (0.911) when paired with Word2Vec embeddings, significantly outperforming other models with the same embedding. This suggests that Word2Vec effectively captures features highly relevant to LR's linear decision boundary. However, its performance drops substantially with TF-IDF. DTs show relatively consistent performance across embeddings, but generally lower accuracy than LR and RF, especially with TF-IDF, and high training times across the board. RF consistently outperforms DTs, showcasing the benefits of bagging in enhancing accuracy and robustness.

Similar to LR, it benefits significantly from Word2Vec embeddings, achieving high accuracy with relatively faster training times compared to DT, second only to LR with Word2Vec. In contrast, SVM consistently underperforms across all embeddings, displaying consistently low accuracy regardless of the embedding method. This suggests that SVM struggles to effectively capture the relevant features for this specific classification problem, with its chosen parameters.

The anomalously low SVM performance, particularly with Word2Vec embeddings (accuracy: 0.331), can be attributed to several factors. First, SVM is known to be sensitive to high-dimensional dense vectors; Word2Vec produces 300-dimensional dense embeddings, which may cause the algorithm to struggle with finding an optimal hyperplane. Second, the max_iter=1000 setting may have been insufficient for convergence given the large dataset size (~490K training instances), meaning the optimizer did not reach a stable solution. Third, SVMs are computationally expensive for large-scale datasets, and the long training times observed (443 seconds with Word2Vec) further support the hypothesis that the model did not fully converge. Future work could explore kernel tuning and increased iteration limits to potentially improve SVM performance on this task.

The unexpected superiority of Word2Vec over FastText in this study can be attributed to the nature of the sentiment analysis task. Sentiment classification relies primarily on whole-word semantic representations, where the overall meaning of a word carries more discriminative power than its constituent subword components. FastText's subword information, while beneficial for morphologically rich languages

in tasks such as named entity recognition or text classification involving rare words, may introduce noise in sentiment-oriented tasks where the emotional polarity is conveyed at the word level rather than the subword level. Consequently, Word2Vec's cleaner word-level representations may align more naturally with the linear decision boundaries learned by Logistic Regression, explaining its superior performance on this dataset.

The high training times for SVM with Word2Vec and FastText further highlight this model's inefficiency for this task. To further investigate model behavior, Fig 2 presents the confusion matrices for LR, using each of the three embedding methods.

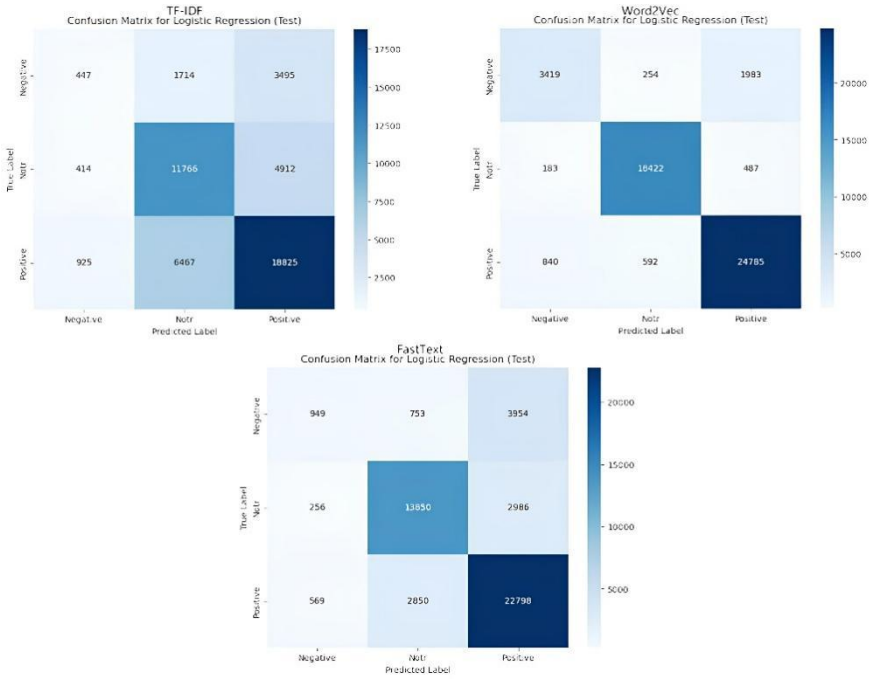

Fig. 2. Confusion matrices for LR with TF-IDF, Word2Vec, and FastText embeddings.

The choice of both embedding method and machine learning model significantly impacts the performance of the text classification system. Word2Vec demonstrates superior performance as an embedding method, producing significantly higher accuracy scores across most models. However, this comes at the cost of increased training time. FastText provides a good compromise between accuracy and speed. Among the machine learning models, LR and RF achieve the highest accuracy, particularly when combined with Word2Vec embeddings. However, SVM consistently underperforms, suggesting that it may not be the most appropriate model for this particular task and dataset. Further investigation could explore hyperparameter tuning for each model to potentially improve performance, particularly for SVM. Furthermore, understanding the characteristics of the dataset and choosing the appropriate evaluation metrics is

critical for a robust and informed model selection. Fig 3 provides a visual comparison of the accuracy achieved by each model for the different embedding techniques.

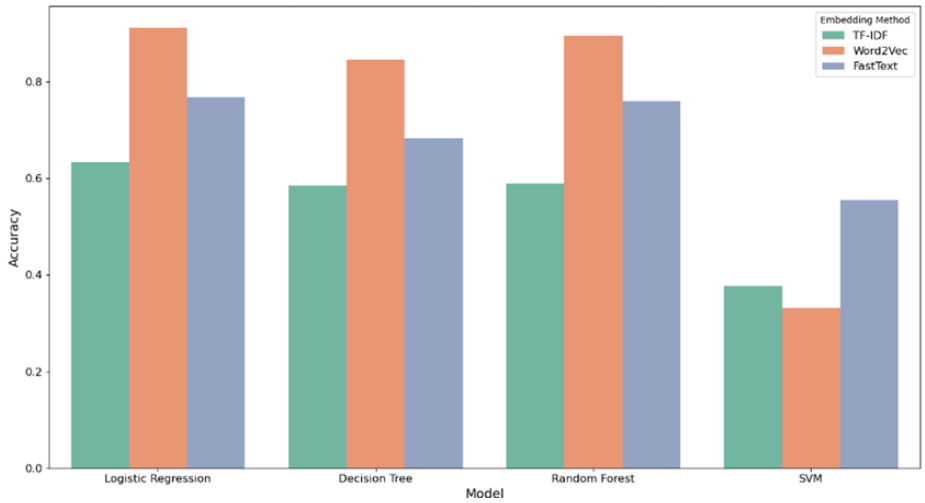

Fig. 3. Accuracy comparison of models across different embedding techniques.

5 Conclusion

This study performed a comprehensive comparative analysis of three prominent word embedding techniques (TF-IDF, Word2Vec and FastText) for Turkish sentiment analysis using four different machine learning models. Our results show that Word2Vec consistently outperforms TF-IDF and is comparable to FastText across multiple classifiers. The superior performance of Word2Vec and FastText, especially compared to the simpler TF-IDF approach, highlights the importance of considering contextual information and subword morphology to achieve high accuracy in sentiment classification for morphologically rich languages. While the context-independent nature of TF-IDF limited its effectiveness, the capacity of Word2Vec and FastText to capture semantic relations and subword structures significantly improved the classification accuracy. When comparing Word2Vec and FastText, Word2Vec performed better. This outcome suggests that for sentiment analysis tasks, word-level semantic representations may be more effective than subword-based approaches, as sentiment polarity is predominantly carried at the word level rather than the subword level. Future research could investigate the effectiveness of incorporating advanced techniques such as attention mechanisms or transducer-based architectures to further improve the performance of sentiment analysis models in Turkish. Furthermore, further research on larger and more diverse datasets may help to verify the generalizability of our findings and improve the selection of the most appropriate embedding techniques for different applications. The limitations of our study, mainly related to the specific dataset and the selected classifiers, should be taken into account when interpreting the results and guiding future research directions.

References

1. Atf, Z., Taherikia, F., & Heidarzadeh Hanzaee, K. (2023). Modeling Brand Engagement in Social Media (Based on Sentiment Analysis and Customer Data). International Journal of Innovation Management and Organizational Behavior (IJIMOB), 3(2), 208-218.
2. Farias, D. H., & Rosso, P. (2017). Irony, sarcasm, and sentiment analysis. In Sentiment analysis in social networks (pp. 113-128). Morgan Kaufmann.
3. Mohd Sofi, S., & Selamat, A. (2023). Aspect Based Sentiment Analysis: Feature Extraction using Latent Dirichlet Allocation (LDA) and Term Frequency - Inverse Document Frequency (TF-IDF) in Machine Learning (ML). Malaysian Journal of Information and Communication Technology (MyJICT), 8(2), 158-168. https://doi.org/10.53840/myjict8-2-102
4. Wu, H., & Yuan, N. (2018, May). An Improved TF-IDF algorithm based on word frequency distribution information and category distribution information. In Proceedings of the 3rd International Conference on Intelligent Information Processing (pp. 211-215)
5. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
6. Sivakumar, S., Videla, L. S., Kumar, T. R., Nagaraj, J., Itnal, S., & Haritha, D. (2020, September). Review on word2vec word embedding neural net. In 2020 international conference on smart electronics and communication (ICOSEC) (pp. 282-290). IEEE.
7. Sharma, A. (2023). Context-aware Sentiment Analysis on Refined Word Embeddings Word2Vec Model. Authorea Preprints.
8. Vasić, D., & Brajković, E. (2018, September). Development and evaluation of word embeddings for morphologically rich languages. In 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM) (pp. 1-5). IEEE.
9. Savelli, C., & Giobergia, F. (2024, September). Enhancing Cross-Lingual Word Embeddings: Aligned Subword Vectors for Out-of-Vocabulary Terms in fastText. In 2024 IEEE 18th International Conference on Application of Information and Communication Technologies (AICT) (pp. 1-6). IEEE.
10. "winvoker/turkish-sentiment-analysis-dataset · Datasets at Hugging Face", 09 Jan 2025, https://huggingface.co/datasets/winvoker/turkish-sentiment-analysis-dataset
11. Demircan, M., Seller, A., Abut, F., & Akay, M. F. (2021). Developing Turkish sentiment analysis models using machine learning and e-commerce data. International Journal of Cognitive Computing in Engineering, 2, 202-207.
12. Aydın, C. R., & Güngör, T. (2021). Sentiment analysis in Turkish: Supervised, semi-supervised, and unsupervised techniques. Natural Language Engineering, 27(4), 455-483.
13. Kemaloğlu, N., Küçüksille, E., & Özgünsür, M. (2021). Turkish sentiment analysis on social media. Sakarya University Journal of Science, 25(3), 629-638.
14. Özdemir, M., & Ortakcı, Y. (2024). Text Classification with Frequency-Based Text Vectorisation Methods for Enhancing Call Centre Efficiency. Current Trends in Computing, 1(2), 122-138.
15. Acikalin, U. U., Bardak, B., & Kutlu, M. (2020, October). Turkish sentiment analysis using Bert. In 2020 28th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
16. Guven, Z. A. (2022). The comparison of language models with a novel text filtering approach for turkish sentiment analysis. ACM Transactions on Asian and Low-Resource Language Information Processing, 22(2), 1-16.

17. Ba Alawi, A., & Bozkurt, F. (2024). Performance Analysis of embedding methods for deep learning-based Turkish sentiment analysis models. Arabian Journal for Science and Engineering, 1-23.
18. Alshammari, Q., & Akyüz, S. (2024). Mental Health on Twitter in Turkey: Sentiment Analysis with Transformers. In Decision Making in Healthcare Systems (pp. 391-402). Cham: Springer International Publishing.
19. Cam, H., Cam, A. V., Demirel, U., & Ahmed, S. (2024). Sentiment analysis of financial Twitter posts on Twitter with the machine learning classifiers. Heliyon, 10(1).
20. Taşar, D. E., Özcan, C., & Koruyan, K. (2022). Autotrain Yaklaşimi İle Duygu Analizi. Proceeding Book, 30.
21. Özdemir, M., & Ortakcı, Y. (2024). An Analysis of Intelligent Turkish Text Classification Models for Routing Calls in Call Centers: A Case Study on the Republic of Türkiye Ministry of Trade Call Center. Sakarya University Journal of Computer and Information Sciences, 7(1), 46-60.
22. Bui, Q. A., Visani, M., Prum, S., & Ogier, J. M. (2011, September). Writer identification using TF-IDF for cursive handwritten word recognition. In 2011 International Conference on Document Analysis and Recognition (pp. 844-848). IEEE.
23. Naeem, M. Z., Rustam, F., Mehmood, A., 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.
24. Alfarizi, M. I., Syafaah, L., & Lestandy, M. (2022). Emotional text classification using tf-idf (term frequency-inverse document frequency) and lstm (long short-term memory). JUITA: Jurnal Informatika, 10(2), 225-232.
25. Alshuraiqi, H. (2020). Improved term frequency inverse document frequency (TF-IDF) method for Arabic text classification. International Journal of Advanced Trends in Computer Science and Engineering, 9(1), 6939–6946.
26. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 26.
27. Jatnika, D., Bijaksana, M. A., & Suryani, A. A. (2019). Word2vec model analysis for semantic similarities in english words. Procedia Computer Science, 157, 160-167.
28. Li, P. (2023). The study for word prediction based on Skip-gram and CBOW model. Theoretical and Natural Science, 18(1), 210–215.
29. Kang, H., & Yang, J. (2020). Performance comparison of Word2vec and fastText embedding models. Journal of Digital Contents Society, 21(7), 1335-1343.
30. Liu, K. (2021). Incorporate Out-of-Vocabulary Words for Psycholinguistic Analysis using Social Media Texts-An OOV-Aware Data Curation Process and a Hybrid Approach (Doctoral dissertation, The Claremont Graduate University).
31. Nasution, N., Nababan, E., & Mawengkang, H. (2024). Comparing LSTM algorithm with word embedding: FastText and Word2Vec in Bahasa Batak-English translation. Proceedings of the International Conference on Information and Communication Technology (ICoICT), 306–313.
32. Das, A. (2024). Logistic regression. In Encyclopedia of Quality of Life and Well-Being Research (pp. 3985-3986). Cham: Springer International Publishing.
33. De Ville, B. (2013). Decision trees. Wiley Interdisciplinary Reviews: Computational Statistics, 5(6), 448-455.
34. Iranzad, R., & Liu, X. (2024). A review of random forest-based feature selection methods for data science education and applications. International Journal of Data Science and Analytics, 1-15.

35. Chandra, M. A., & Bedi, S. S. (2021). Survey on SVM and their application in image classification. International Journal of Information Technology, 13(5), 1-11.