Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.5, October 2025, pp. 2270~2277
e-ISSN: 2548-6861

Comparison of Text Vectorization Methods for IMDB Movie Review Sentiment Analysis Using SVM

Rifqi Mulyawan [1]*, Husni Naparin [2]*, Wifda Muna Fatihia [3]*
* Information Technology, UIN Antasari Banjarmasin
rifqi.mulyawan@uin-antasari.ac.id [1] parinhusnina@uin-antasari.ac.id [2], wifdamunafatihia@uin-antasari.ac.id [3]

Article Info

Article history:

Received 2025-07-22
Revised 2025-09-01
Accepted 2025-09-03

Keyword:

Bag of Words,
Doc2Vec,
TF-IDF,
Support Vector Machine,
Word2Vec.

ABSTRACT

Sentiment Analysis is a scientific study in the field of Machine Learning that focuses on classifying opinions expressed in text. IMDb is a platform widely used to provide information and share viewpoints among moviegoers worldwide, where audience reactions often serve as a benchmark for a movie's success. This research aims to classify positive and negative sentiments by applying and evaluating the effectiveness of Support Vector Machine (SVM) with four different feature representation methods: (a) Bag of Words (BoW), (b) TF-IDF, (c) Word2Vec, and (d) Doc2Vec. After preprocessing the textual data, each method was employed to extract features for model training. The experimental results demonstrate that the combination of SVM with Word2Vec achieved the best overall performance with an F1-Score of 0.8607 and an Accuracy of 0.8607, while also being the fastest in training time (75.0s). In comparison, BoW reached an F1-Score of 0.8219, TF-IDF achieved 0.8520, and Doc2Vec obtained 0.8440. These findings highlight that Word2Vec provides the most effective feature representation for sentiment classification using SVM in this study.

I. INTRODUCTION

Over the past ten years, the rapid growth of the web has transformed how people share their thoughts and opinions. Nowadays, user opinions are widespread and accessible across numerous online platforms, including news websites, personal blogs, social media, discussion boards, and review sites. This shift has led to an overwhelming surge of user-generated content flooding the internet. Catching popular opinion about any social events, movement of national politics, company technique, advertising projects, and also product preferences amasses increasing interest from the clinical area (for tremendous open difficulties), and also from the business globe (for notable advertising failures for feasible market predictions of finance) [1].

Today, assessing and sharing experiences concerning products and services is a prevailing practice. Opinion mining has gained significant focus from the community study in recent years because of its big-data relevance [2], many challenging research issues, and useful applications in both commerce and the academic community.

Sentiment analysis, often referred to as opinion mining, involves the process of identifying, classifying, and extracting subjective information and opinions related to a specific subject. A central challenge in this area lies in determining sentiment polarity—categorizing opinions as positive or negative—which may appear at the document, sentence, or feature level. This field draws upon techniques from data mining and Natural Language Processing (NLP) to uncover sentiments expressed in textual content, primarily found online [3]. Numerous methods and strategies have been proposed by researchers to improve the accuracy and effectiveness of sentiment analysis tasks. The general concept of categorizing views from messages is to designate "positive" or "negative" (or even "neutral") labels for blocks of text (papers, sentences, reviews, etc.).

Research [4] was the very first to propose sentiment classification utilizing machine learning designs. They analyzed the Naïve Bayes, Max Entropy, and SVM versions for view evaluation on unigrams as well as bigrams of information. In their experiment, SVM matched with unigrams generated very good results.

Mullen and Collier [5] utilized Support Vector Machines (SVMs), and the feature set for document representation was expanded by incorporating sentiment indicators derived from a variety of external sources. Both of them introduced functions based upon Osgood's Concept of Semantic Differentiation [6] By leveraging WordNet, the values associated with adjectives—such as their effectiveness, intensity of action, and evaluative meaning—were extracted, alongside the application of Turney's semantic orientation technique [4].

Support Vector Machine is superior in cases with datasets that are not too large in number. In addition, SVM is effective on data represented as text, as it can find the optimal hyperplane that separates the classes with maximum margin [7]. The Naïve Bayes algorithm is faster, but its accuracy is often lower due to the assumption of word independence [8]. Research [9] proved that the SVM algorithm tested in the case of sentiment analysis of IMDb review data achieved an accuracy value of 86.5% with a precision value of 90.67% and a recall value of 91.62%. Our research chose SVM as the classification algorithm by considering some of the previous studies mentioned earlier. From the aspect of accuracy value, SVM provides a high accuracy value for text data, making it suitable for data with thousands of words or significant features. In addition, the IMDb dataset contains varied opinions, but SVM can ignore irrelevant words, reducing noise in the dataset processing.

As with other classification algorithms, the SVM model operates on numerical data, meaning that every record in the training set must undergo a vectorization process. Although the basic BoW method often yields suboptimal results, its improved counterpart—TF-IDF—offers better performance by handling stop words and assigning a significance score to each term [10]. However, both methods fail to capture semantic features and ignore the sequence of words within a phrase. To address these limitations, more advanced models for numerical document representation—such as Word2Vec [11] and Doc2Vec [12] —can be employed. Although both models are computed in a similar manner, they offer slightly different advantages. Word2Vec produces word embeddings that are highly adaptable and transferable across various domains, whereas Doc2Vec is more closely aligned with the domain of the training data, thus providing deeper contextual understanding. While Word2Vec offers greater flexibility across datasets, Doc2Vec tends to perform better when dealing with lengthy documents [13]. Nonetheless, both neural network-based models are generally expected to outperform traditional TF-IDF representations [14].

BoW, TF-IDF, Word2Vec, and Doc2Vec represent the two main generations of text representation techniques so that the research results can directly compare the performance of traditional and modern approaches [15]. BoW and TF-IDF are traditional approaches (frequency-based) that are fast, simple, and suitable for baseline. Word2Vec and Doc2Vec are modern approaches (neural embeddings) capable of understanding context and semantic relations [16].

To validate our experiments, we adopt the widely used "IMDb Movie Reviews" dataset and analyze whether deep semantic representations outperform conventional heuristic techniques like BoW and TF-IDF. We also investigate whether using alternative classification algorithms may further improve model performance. In this paper, we compare and evaluate four methods for opinion mining using the Support Vector Machine (SVM) algorithm on the IMDb dataset. IMDb is a site that shares rating and review information about films, TV programs, home videos, games, streaming videos, and other celebrity content [17]. For evaluation, we utilized IMDb's movie reviews dataset retrieved from Kaggle. This study explores four different text representation techniques: (a) Bag of Words, (b) TF-IDF, (c) Word2Vec, and (d) Doc2Vec. Our primary goal is to develop and evaluate SVM model configurations that aim to improve both classification efficiency and effectiveness. As a first step, we assess which text representation method yields the most reliable performance while maintaining a reasonable computational cost. To do this, we compare traditional methods such as Bag of Words and TF-IDF with more modern neural network-based representations like Word2Vec and Doc2Vec, which stem from deep learning approaches. Deep learning has increasingly become a valuable tool for decision-makers, offering improvements in solving new challenges or enhancing classical algorithms [18].

Implementation of different mixes of pre-processing approaches can boost viewpoint classification and results. Methods like BoW can link a message with a vector, revealing the number of occurrences of each selected word in the training corpus. It is frequently suggested to carry out a comprehensive and systematic variety of pre-processing strategies incorporated with category experiments, given that it contributes to boosting the accuracy. According to [19], for all the tested datasets, there was always a minimum of one mix of fundamental pre-processing techniques that could be advised to dramatically improve the text classification by using BoW representation.

The remainder of this paper is structured as follows: the next section discusses related work in the field, followed by an explanation of the SVM model, implementation details, vectorization strategies, experimental setup, and the results obtained. The last section is devoted to experiments and provides a brief verdict of our paper.

II. LITERATURE REVIEW

Support Vector Machines (SVMs) have earned a stellar reputation in machine learning, consistently hailed as one of the most precise and reliable discriminative classifiers. Their theoretical backbone is rooted in the Structural Risk Minimization (SRM) principle, a cornerstone concept from computational learning theory that seeks to strike an optimal balance between model complexity and generalization performance. By rigorously minimizing the upper bound on the true classification error, SVMs excel at delivering robust

predictions even in high-dimensional spaces. Beyond their empirical prowess, these models are celebrated for their mathematical elegance, offering a transparent framework that facilitates in-depth theoretical scrutiny and intuitive interpretation. Their versatility and interpretability make SVMs a favorite among researchers and practitioners, bridging the gap between abstract theory and real-world applicability [20].

The proposed methodologies for extracting and categorizing opinions from textual data can be broadly classified into two dominant paradigms: machine learning (ML)-based techniques and lexicon-driven approaches. Machine learning methods leverage advanced algorithms to automatically identify patterns and sentiments, often requiring large annotated datasets for training. In contrast, lexicon-based methods rely on predefined dictionaries of sentiment-laden words and linguistic rules, offering greater interpretability but sometimes lacking adaptability. While ML approaches excel in handling complex, context-dependent language, lexicon-based systems provide a more transparent and rule-governed framework, making them suitable for domains with well-defined sentiment expressions. This dichotomy highlights the trade-offs between scalability and explainability in opinion mining tasks [20].

The first approach frames opinion mining as a traditional text classification task, employing machine learning algorithms that harness a combination of syntactic structures (e.g., word order, grammar) and linguistic features (e.g., sentiment-laden terms, negation patterns) to discern subjective expressions. These models learn from annotated datasets, enabling them to generalize and predict sentiment labels for unseen text accurately.

In contrast, the second approach adopts a lexicon-based strategy, relying on curated dictionaries where words and phrases are pre-assigned polarity values (e.g., positive, negative, neutral intensity scores). This method applies rule-based systems to aggregate sentiment indicators, often incorporating modifiers (e.g., "very," "not") to adjust final sentiment scores. While less data-dependent than ML, lexicon-based techniques require meticulous lexical resource construction and may struggle with domain-specific or context-dependent language.

Supervised machine learning techniques, including Support Vector Machines (SVM), fundamentally rely on labeled training datasets encompassing positive and negative sentiment examples. These datasets empower the SVM to determine an optimal decision boundary—formally termed a hyperplane—within an n-dimensional feature space, thereby maximizing the separation margin between the two classes. The most critical data points, or support vectors, reside closest to this hyperplane and directly influence its positioning. A remarkable property of SVMs is their robustness to non-support vectors: even if documents distant from the boundary are omitted during training, the model's performance remains unaffected mainly, as the support vectors solely define the hyperplane. This efficiency underscores SVMs' capability to generalize well, even with sparse but strategically selected data [21].

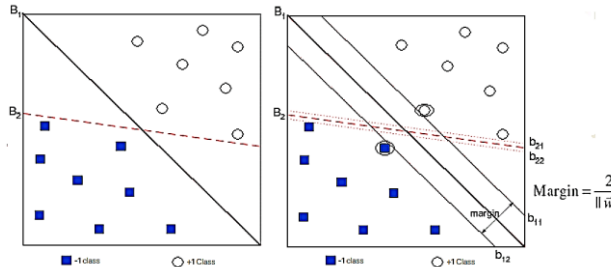

Figure 1. SVM Hyperplane Creation

Support Vector Machines (SVMs) remain among the most widely used classification models, with applications spanning text processing, image categorization, and handwritten character recognition. The original SVM formulation was introduced in 1993, based on theoretical foundations established roughly three decades earlier by Vapnik and Chervonenkis. At its core, the SVM model seeks to separate data into two primary classes using a linear classifier.

As we can see above (Fig. 1), let the input data be denoted as $x \in \mathbb{R}^d$, where each instance is assigned a label $y_i \in \{-1, +1\}$ for i = 1, 2, ..., n, with n representing the total number of samples. If the two classes are linearly separable in a d-dimensional space, the decision boundary—also known as a hyperplane—can be expressed as [22]:

$$w^t x + b = 0 \tag{1}$$

Because the hyperlane divide two locations based on each other class, so each x_i the sample that belongs -1 dan $+1$ class will not fulfill the equation:

For $y_i = -1$:

$$w.x_i + s \leq -1 \tag{2}$$

For $y_i = +1$:

$$w.x_i + s \leq +1 \tag{3}$$

So that the two equal margins can be calculated by subtracting equations (2) and (3), and then [23] we can get the following equation:

$$w.(x_1 - x_2) = 2 \tag{4}$$

$$\left[\frac{w}{\|w\|}(x_1 - x_2)\right] = \frac{2}{\|w\|} \tag{5}$$

The fundamental objective of margin maximization in Support Vector Machines revolves around identifying the most substantial possible separation boundary between classes. This process involves carefully calibrating the spatial gap separating the decision hyperplane from its closest neighboring observations - those critical data specimens we

refer to as support vectors. From a computational standpoint, we can transform this geometric pursuit into an elegant mathematical formulation where we seek to reduce the magnitude of the weight vector ‖w‖ to its minimal possible value. This minimization must occur within strictly defined parameters that guarantee accurate classification of every training instance in our dataset.

When we express this optimization challenge in its purest mathematical form, we arrive at the beautifully simple yet profoundly significant expression $\frac{2}{\|w\|}$ - a representation that captures the inverse relationship between the weight vector's magnitude and the margin size we aim to expand. This problem can be expressed as a quadratic programming (QP) task, aiming to determine the minimum point of a given equation:

$$\min_{w} \tau(w) = \frac{1}{2} \|w\|^2 \quad (6)$$

With a constraint that must be satisfied according to the following inequality:

$$y_i \, (x_i . w + s) - 1 \leq 0, \forall i \quad (7)$$

To solve the (4) and (5) problems, we can use the *Lagrange multiplier* with the formula:

$$L(w, s, a) = \frac{1}{2} \|w\|^2 - \sum_{i=l}^{l} a_i (y_i ((x_i . w + s) - 1)); (i = 1,2, \dots, n) \quad (8)$$

Where a_i is *lagrange multiplier* that has a bigger value than 0, $a_i \leq 0$. Take notice of the nature that the *optimal gradient* $L = 0$, so (8) we can modify it as a maximization problem which only consists of a_i as following equating:

Maximized:

$$\sum_{i=l}^{l} a_i - \frac{1}{2} \sum_{i,j=l}^{l} a_i a_j y_i y_j x_i x_j \quad (9)$$

Fulfilled equation:

$$a_i \leq 0 (i = 1,2, \dots, l) \quad \sum_{i=l}^{l} a_i y_i = 0 \quad (10)$$

The results of the above calculations will produce a positive *lagrange multiplier* (a_i) where later the correlated data with a_i It is called a *support* vector [24].

III. METHODS

A. Dataset

This study utilizes the IMDb movie reviews dataset obtained from Kaggle, which originally consists of 50,000 samples with three columns: review, sentiment, and label. The dataset is evenly distributed, comprising 25,000 positive reviews (50.0%) and 25,000 negative reviews (50.0%). For the purposes of this research, a subset of 15,000 samples was selected. The data was subsequently divided into a training set of 10,500 samples and a test set of 4,500 samples. Figure 2 shows the distribution of review lengths by character length.

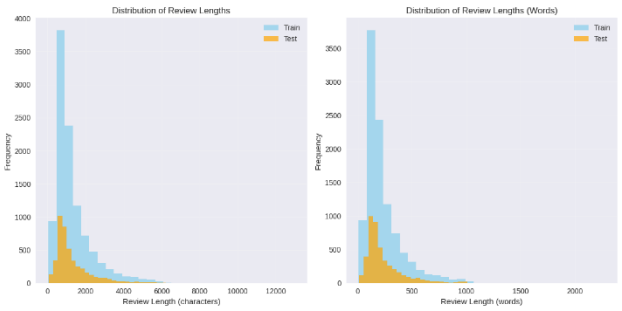

Figure 2. Review Length Distributions

B. Data Preprocessing

Text represents a highly unstructured form of data that requires substantial preprocessing before becoming suitable for analysis. The complete process of cleaning and standardizing textual data - removing noise and preparing it for examination - is collectively known as text preprocessing. As illustrated in Figure 3 and Figure 4's word cloud visualization, raw text data typically contains numerous uninformative elements that obscure meaningful patterns. Our preprocessing pipeline consists of two primary stages: (1) data inspection and (2) data cleaning. The cleaning phase systematically removes elements irrelevant for review classification, including punctuation, numerical values, and special characters.

Additionally, we eliminate short function words (such as "him," "all," and "the") that contribute minimal semantic value, as their frequent appearance in the word cloud (Figure 3 and Figure 4) demonstrates their limited analytical utility. The data then undergoes normalization, where derived word forms (e.g., "like," "liking," "likable") are reduced to their base forms ("like") through lemmatization before final tokenization. This comprehensive preprocessing workflow, visually evidenced by the transformation from Figure 3 and Figure 4's initial word cloud to the cleaned version, ensures our text data achieves optimal quality for subsequent analysis. Figure 3 shows the positive word cloud movie review vocabulary, while Figure 4 illustrates the negative word cloud of movie review vocabulary.

Figure 3. Positive Word Cloud Movie Review Vocabulary

Figure 4. Negative Word Cloud of Movie Review Vocabulary

C. Modeling

Most classification algorithms operate on numerical data, which requires textual inputs to be transformed into vector representations before classification. Before this vectorization step, raw text undergoes a preprocessing pipeline that includes cleaning (removing punctuation marks, numbers, and special characters) and lemmatization. In addition, irrelevant words are filtered out through weighting schemes such as TF-IDF or other frequency-based metrics. Figure 5 illustrates the block diagram of our text classification pipeline, starting from the IMDb dataset of 15,000 reviews, followed by preprocessing, feature extraction using BoW, TF-IDF, Word2Vec, or Doc2Vec, and finally classification using machine learning algorithms (e.g., SVM).

The dataset was split into a training set (70%, 10,500 samples) and a test set (30%, 4,500 samples) using stratified sampling to preserve the original sentiment distribution. Four approaches were employed for feature extraction. The Bag of Words (BoW) method provides a simple frequency-based representation of words, while TF-IDF applies weighted word frequencies to account for document-level importance. Word2Vec generates distributed vector representations that capture semantic similarity between words, and Doc2Vec extends this idea by producing document-level embeddings using the Distributed Memory (DM) and Distributed Bag of Words (DBOW) architectures.

The extracted features were then used to train machine learning algorithms, primarily focusing on Support Vector Machine (SVM with Linear kernel, C=1, probability=True, random_state=42). We implemented a comprehensive evaluation procedure measuring Accuracy, Precision, Recall, and F1-score to ensure robust evaluation. Additionally, training time was recorded to assess computational efficiency.

An important step in the methodology was threshold optimization for probabilistic models. By testing thresholds from 0.3 to 0.7, the system selected the optimal cutoff point that maximized the F1-Score. It is ensured that the classification results were not biased toward a fixed threshold (0.5) but reflected the best achievable trade-off between Precision and Recall.

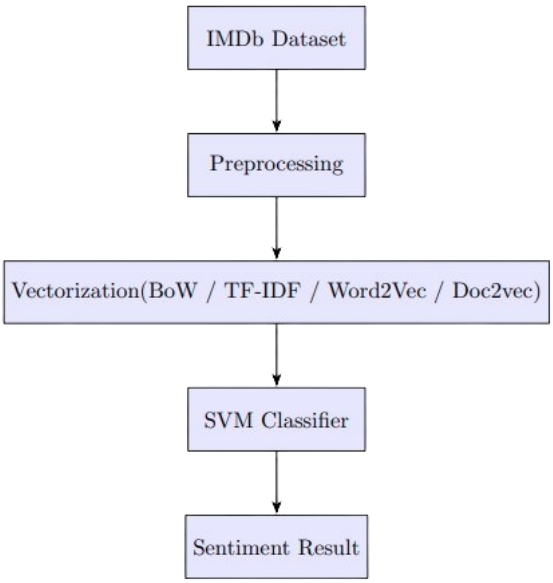

Figure 5. Block Diagram of Text Classification Pipeline

Word2Vec creates meaningful word representations by training a specialized neural network to predict word contexts within a given text corpus. This innovative approach utilizes two distinct but related neural architectures that differ primarily in their prediction direction. The first architecture, Continuous Bag of Words (CBOW), functions by predicting a target word from its surrounding context words, operating similarly to traditional N-Gram models but with greater neural sophistication. The second architecture, known as Skip-Gram (SG), reverses this prediction logic by using a single input word to predict its likely context words.

Despite their different prediction approaches, both models employ an identical fundamental structure: they contain a projection layer (sometimes referred to as a hidden layer) with linear activation that serves as the actual word embedding space, followed by an output layer that uses a non-linear activation function (typically softmax or a sampling-based alternative) to generate probability distributions over the vocabulary. This elegant architecture allows Word2Vec to efficiently transform sparse word representations into dense, low-dimensional vectors that remarkably capture semantic and syntactic word relationships [25].

The model's weight optimization employs stochastic gradient descent (SGD) enhanced by backpropagation, a computationally efficient approach that iteratively adjusts weights across both the output and hidden (projection) layers. This training process transforms textual data into numerical representations through two primary vectorization approaches: concatenation, which preserves sequential information by combining word vectors end-to-end, or aggregation (typically averaging), which creates a composite representation of multiple words. During this transformation, the document corpus manifests in two distinct mathematical spaces: at the output layer, words are distributed across a two-

dimensional probability space representing the vocabulary, while the hidden/projection layer compresses each word's semantic essence into a single, dense vector embedding. This dual-space representation enables the model to maintain the discrete, observable characteristics of words in the output layer while capturing their continuous, latent relationships in the embedding space. The backpropagation mechanism carefully coordinates updates between these layers, ensuring the emerging embeddings preserve both syntactic and semantic word relationships as reflected in their original textual contexts [26].

Building upon Word2Vec's successful architecture, the Doc2Vec framework extends the concept of word embeddings to entire documents through two specialized neural network architectures: Distributed Memory (DM) and Distributed Bag of Words (DBOW). Both models inherit and adapt the fundamental patterns established by Word2Vec while introducing document-level representations. The DM model, analogous to Word2Vec's CBOW architecture, generates predictions by analyzing a context window of words supplemented with a unique document identifier. This implementation maintains two separate weight matrices: matrix W for word vectors and matrix D for document vectors. During prediction, the model combines the document vector (retrieved as a column from matrix D) with context word vectors (from matrix W) through either averaging or concatenation operations, creating a rich composite representation that captures both word-level and document-level semantics.

In contrast, the DBOW model adopts Skip-Gram's predictive approach but operates at the document level, using a document identifier as input to predict its constituent words. This architecture proves particularly efficient as it does not require word vector storage during inference. After the training, the learned document vectors (columns in matrix D) serve as ready-to-use numerical representations of entire documents, eliminating the need for additional processing or feature engineering. These document embeddings effectively capture thematic content and contextual information, making them immediately suitable for downstream tasks like document classification, clustering, or similarity analysis [13].

The pre-processed text data is transformed into model-ready features, initially employing the Bag-of-Words (BoW) technique, followed by Term Frequency–Inverse Document Frequency (TF-IDF). To capture richer semantic information, we adopt Word2Vec embeddings, which offer a refined method of expressing words as vectors, compressing their original high-dimensional representations into lower-dimensional ones while maintaining contextual similarity within the dataset. Accordingly, we train a Word2Vec model on our dataset to produce dense vector forms for each distinct word. Given that our dataset comprises full review texts rather than isolated words, we aggregate the word vectors to generate a unified representation for each review. In the case of Doc2Vec modeling, every tokenized review must first be assigned a distinct identifier before constructing the corresponding feature vectors.

D. Result and Evaluation

For evaluating these four different approaches, we use the F1 score (based on weighted average of precision and recall) as a metric evaluation that consists of important components like (1) True Positive (TP) that correctly predict positive samples (2) True Negative (TN) that correctly predict negative review samples (3) False Positive (FP) that counted when a sample is not on actual class but predicted as the class (4) False Negative (FN) is an opposite of FP.

We employed the F1-Score as the primary evaluation metric, which balances precision and recall to provide a reliable measure of classification performance. Accuracy, Precision, Recall, and Training Time were also recorded to provide a more comprehensive performance comparison.

The results are summarized in Table 1, which presents the top-performing models based on the F1 Score. Among all approaches, the SVM (linear) with Word2Vec representation consistently achieved the best performance, with an F1 Score of 0.8607, Accuracy of 0.8607, and the fastest training time of 75.0 seconds.

TABLE I
BEST PERFORMING MODELS (BY F1-SCORE)

Feature Method	Accuracy	Precision	Recall	F1-Score	Training Time (s)
Word2Vec	0.8607	0.8607	0.8607	0.8607	75.0
TF-IDF	0.8520	0.8521	0.8520	0.8520	285.5
Doc2Vec	0.8442	0.8458	0.8442	0.8440	569.3
BoW	0.8220	0.8224	0.8220	0.8219	390.3

Based on the test results, the four vectorization methods have their limitations. Bag of Words (BoW) shows the lowest performance because it only relies on word frequency without considering context or semantic relationships, and it produces a less efficient and very high data dimension. TF-IDF is slightly better as it considers the weight of words based on distribution, but it still does not understand contextual meaning and requires longer training time due to the sparse vector representation. Doc2Vec represents the document to capture more global information. However, its performance results are unstable on small datasets, parameterization is more complex, and training time is the longest. Meanwhile, Word2Vec performs best with the highest accuracy and F1-score and the most efficient training time. However, it is still limited because it only represents words at the local semantic level without maintaining the full context between sentences.

E. Experimental Result

Six different machine learning algorithms were used to evaluate models using Word2Vec features. The results are summarized in Table 4, ranked by their F1 Score performance. Overall, the experiments demonstrate that SVM (RBF) with Word2Vec features provides the best performance, with an F1 Score of 0.8593 and an Accuracy of

0.8593. In contrast, the Decision Tree with Word2Vec yielded the lowest performance, with an F1 Score of 0.7011.

TABLE II
COMPARISON OF EXPERIMENTAL RESULTS

Algorithm	Accuracy	Precision	Recall	F1-Score	Training Time (s)
SVM (RBF)	0.8593	0.8593	0.8593	0.8593	99.6
Logistic Regression	0.8542	0.8543	0.8542	0.8542	0.149
Gradient Boosting	0.8327	0.8329	0.8327	0.8326	123.3
Random Forest	0.8169	0.8178	0.8169	0.8167	15.3
KNN	0.7802	0.7891	0.7802	0.7782	0.015
Decision Tree	0.7011	0.7012	0.7011	0.7011	4.7

In terms of efficiency, KNN recorded the fastest training time (0.015s), followed closely by Logistic Regression (0.149s), while Gradient Boosting and SVM required longer training times. This trade-off between accuracy and efficiency indicates that Logistic Regression with Word2Vec could be a practical alternative when computational resources are limited.

IV. CONCLUSION

This study comprehensively evaluates four distinct text representation approaches to determine their relative effectiveness for sentiment analysis tasks. To ensure robust evaluation, we systematically compare each representation method using multiple classification algorithms, including logistic regression and random forest classifiers.

Our experiments on the benchmark "IMDb Movie Reviews" dataset demonstrate that the Word2Vec embedding approach consistently outperforms alternative methods, achieving superior classification accuracy. While traditional techniques like Bag-of-Words (BoW) and TF-IDF offer computational simplicity and remain popular baseline methods, our analysis reveals their inherent limitations. These approaches fail to preserve syntactic relationships between words and cannot capture deeper semantic meanings. Our results suggest that these shortcomings can be effectively addressed through distributed representation methods like Word2Vec and Doc2Vec, which employ learned weight matrices to encode richer linguistic information.

Our findings indicate that traditional methods can remain competitive in specific scenarios. Specifically, BoW and TF-IDF representations yielded comparable performance to Doc2Vec when paired with random forest classifiers, suggesting that well-configured traditional approaches can sometimes rival more sophisticated modern techniques. This observation underscores an important insight in natural language processing: state-of-the-art models do not universally dominate simpler methods, and optimal approach selection depends on specific use cases and implementation quality.

For future research directions, we identify valuable opportunities to enhance traditional representation methods, particularly by developing hybrid approaches that combine their computational efficiency with aspects of semantic awareness. Further investigation into optimal configuration strategies for classical methods could yield significant practical benefits, especially in resource-constrained environments.

REFERENCES

[1] A. Salinca, "Business Reviews Classification Using Sentiment Analysis," *Proceedings - 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2015*, pp. 247–250, 2016, doi: 10.1109/SYNASC.2015.46.

[2] N. M. Sharef, H. M. Zin, and S. Nadali, "Overview and future opportunities of Sentiment Analysis approaches for big data," *Journal of Computer Science*, vol. 12, no. 3, pp. 153–168, 2016, doi: 10.3844/jcssp.2016.153.168.

[3] M. Taboada, "Sentiment Analysis: An Overview from Linguistics," *Annu Rev Linguist*, vol. 2, no. September, pp. 325–347, 2016, doi: 10.1146/annurev-linguistics-011415-040518.

[4] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," 2002, doi: 10.3115/1118693.1118704.

[5] T. Mullen and N. Collier, "Incorporating topic information into sentiment analysis models," pp. 25-es, 2004, doi: 10.3115/1219044.1219069.

[6] C. E. Osgood, G. J. Suci, and P. H. Tannenbaum, "The Measurement of Meaning [by] Charles E. Osgood, George J. Suci [and] Percy H. Tannenbaum," 1964.

[7] C. K. Wang, "Sentiment Analysis Using Support Vector Machines, Neural Networks, and Random Forests," 2023, pp. 23–34. doi: 10.2991/978-94-6463-300-9_4.

[8] D. Subedi, Nabin Lamichhane, and N. Subedi, "Sentiment Analysis of IMDb Movie Reviews Using SVM and Naive Bayes Classifier," *Journal of Engineering and Sciences*, vol. 4, no. 1, pp. 56–68, May 2025, doi: 10.3126/jes2.v4i1.70138.

[9] G. Cahyani, W. Widayani, S. D. Anggita, Y. Pristyanto, I. Ikmah, and A. Sidauruk, "Klasifikasi Data Review IMDb Berdasarkan Analisis Sentimen Menggunakan Algoritma Support Vector Machine," *Jurnal Media Informatika Budidarma*, vol. 6, no. 3, p. 1418, Jul. 2022, doi: 10.30865/mib.v6i3.4023.

[10] B. Das and S. Chakraborty, "An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation," 2018.

[11] S. Al-Saqqa and A. Awajan, "The Use of Word2vec Model in Sentiment Analysis: A Survey," *PervasiveHealth: Pervasive Computing Technologies for Healthcare*, pp. 39–43, 2019, doi: 10.1145/3388218.3388229.

[12] G. Liu and X. Wu, "Using collaborative filtering algorithms combined with Doc2Vec for movie recommendation," *Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2019*, no. Itnec, pp. 1461–1464, 2019, doi: 10.1109/ITNEC.2019.8729076.

[13] L. Q. Trieu, H. Q. Tran, and M. T. Tran, "News classification from social media using Twitter-based Doc2Vec model and automatic query expansion," *ACM International Conference Proceeding Series*, vol. 2017-Decem, pp. 460–467, 2017, doi: 10.1145/3155133.3155206.

[14] C. Z. Liu, Y. X. Sheng, Z. Q. Wei, and Y. Q. Yang, "Research of Text Classification Based on Improved TF-IDF Algorithm," *2018 IEEE International Conference of Intelligent Robotic and Control Engineering, IRCE 2018*, no. 2, pp. 69–73, 2018, doi: 10.1109/IRCE.2018.8492945.

[15] J. Zhou, Z. Ye, S. Zhang, Z. Geng, N. Han, and T. Yang, "Investigating response behavior through TF-IDF and Word2vec text analysis: A case study of PISA 2012 problem-solving process data," *Heliyon*, vol. 10, no. 16, Aug. 2024, doi: 10.1016/j.heliyon.2024.e35945.

[16] D. Dessi, R. Helaoui, V. Kumar, D. R. Recupero, and D. Riboni, "TF-IDF vs Word Embeddings for Morbidity Identification in Clinical Notes: An Initial Study," Jun. 2021, doi: 10.5281/zenodo.4777594.

[17] K. Kumar, B. S. Harish, and H. K. Darshan, "Sentiment Analysis on IMDb Movie Reviews Using Hybrid Feature Extraction Method," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 5, p. 109, 2019, doi: 10.9781/ijimai.2018.12.005.

[18] Z. Gharibshah, X. Zhu, A. Hainline, and M. Conway, "Deep Learning for User Interest and Response Prediction in Online Display Advertising," *Data Sci Eng*, vol. 5, no. 1, pp. 12–26, 2020, doi: 10.1007/s41019-019-00115-y.

[19] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation," *PLoS One*, vol. 15, no. 5, pp. 1–22, 2020, doi: 10.1371/journal.pone.0232525.

[20] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "Journal of Advances in Information Technology," *Journal of Advances in Information Technology*, vol. 1, no. 1, p. 1, 2010.

[21] H. Brücher, G. Knolmayer, and M.-A. Mittermayer, "Document Classification Methods for Organizing Explicit Knowledge," *CiteSeer*, vol. 41, no. 140, pp. 1–26, 2002.

[22] W. Bourequat and H. Mourad, "Sentiment Analysis Approach for Analyzing iPhone Release using Support Vector Machine," *International Journal of Advances in Data and Information Systems*, vol. 2, no. 1, pp. 36–44, 2021, doi: 10.25008/ijadis.v2i1.1216.

[23] D. Setyawan and E. Winarko, "Analisis Opini Terhadap Fitur Smartphone Pada Ulasan Website Berbahasa Indonesia," *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, vol. 10, no. 2, p. 183, 2016, doi: 10.22146/ijccs.17485.

[24] S. Vijayarani, M. J. Ilamathi, M. Nithya, A. Professor, and M. P. Research Scholar, "Preprocessing Techniques for Text Mining -An Overview," vol. 5, no. 1, pp. 7–16.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pp. 1–12, 2013.

[26] X. Rong, "word2vec Parameter Learning Explained," pp. 1–21, 2014.