Advancing News Text Classification: A Comparative Analysis of Deep Neural Network Models

Nidal Al Said

College of Mass Communication, Ajman University, Ajman, United Arab Emirates
Email: n.alsaid@ajman.ac.ae
*Corresponding author

***Abstract*—The motivation behind this study is to enhance news text classification by comparing deep neural network models. The problem addressed is the challenge of identifying optimal combinations of text processing methods and vector representations to achieve high accuracy, precision, and F1-Scores. The study aims to identify optimal algorithms and text vector representation combinations to achieve the highest accuracy, precision, and F1-Score performance. The approach involves using the 20NEWS dataset and applying a Bidirectional Long Short-Term Memory-Convolutional Neural Network (BL-CNN) architecture integrated with various pre-trained word embeddings such as Word2Vec, FastText, and GloVe. The results indicate that the BL-CNN model with Word2Vec achieves an accuracy of 92.21%, surpassing other combinations. Notably, the Bidirectional Long Short-Term Memory-Convolutional Neural Network-Bidirectional Long Short-Term Memory (BL-CNN-BL) model with FastText achieves an impressive accuracy of 99.32%, highlighting its superiority in text-processing tasks. The findings underscore the effectiveness of specific vector representations in improving classification performance. In conclusion, the proposed models, particularly the BL-CNN-BL with FastText, demonstrate significant potential for accurate and efficient news text classification, with applications in detecting fake news.**

***Keywords*—Bidirectional Long Short-Term Memory (Bi-LSTM), Bidirectional Long Short-Term Memory-Convolutional Neural Network-Bidirectional Long Short-Term Memory (BL-CNN-BL), Convolutional Neural Network (CNN), machine learning, news classification**

I. Introduction

Over the past decade, substantial improvements have transpired in the realms of computer vision, speech recognition, and machine translation, both in research and practical applications. These advancements are largely attributed to recent achievements in neural network models, often characterized by numerous hidden layers known as deep architectures [1, 2]. Presently, there is a growing initiative to employ these neural methods in the information retrieval community. This offers a chance to advance the boundaries of current technologies and potentially achieve groundbreaking results, mirroring successes observed in other fields.

Information retrieval can manifest in various forms, encompassing textual queries expressed by users through keyboards, selection of suggested queries, voice recognition, images, and even implicit needs [3]. Search systems may consider user history, physical location, temporal changes in information, or other contextual factors when ranking results. They may also assist users in formulating their intentions (e.g., through query auto-completion or query suggestions) and (or) extract concise summaries of results for more convenient analysis [4].

This study investigated the effectiveness of different machine learning architectures for text classification tasks, particularly focusing on the 20NEWS dataset. Our findings reveal that the combination of Bidirectional Long Short-Term Memory (Bi-LSTM) and Convolutional Neural Networks (CNN) architectures, when paired with pre-trained word embeddings such as Word2Vec, FastText, and GloVe, offer varied performances across different evaluation metrics. The most contributions of the air study are: 1) The study demonstrates that the Bidirectional Long Short-Term Memory Convolutional Neural Network Bidirectional Long Short-Term Memory (Bi-LSTM-CNN-Bi-LSTM) architecture, particularly when combined with FastText embeddings, significantly outperforms other combinations, achieving an accuracy of 99.32%. This result contrasts with previous works where traditional CNN or LSTM models were used independently, indicating that the bidirectional processing of sequential data, combined with the hierarchical feature extraction of CNNs, offers substantial improvements in accuracy; 2) The study highlights that different pre-trained embeddings exhibit varying degrees of effectiveness. Notably, FastText embeddings, which capture subword information, provided superior performance in the CNN-Bi-LSTM architecture. This finding suggests that the contextual richness of FastText can be particularly advantageous in text classification tasks, surpassing the commonly used Word2Vec and GloVe in certain configurations. This contrasts with prior research where Word2Vec was often preferred for text classification.

Manuscript received June 20, 2024; revised July 9, 2024; accepted November 4, 2024; published May 15, 2025.

The central theme of this work is to enhance text classification performance by exploring and optimizing the integration of Bi-LSTM and CNN architectures with various pre-trained word embeddings. The study aims to determine the most effective combination of these architectures and embeddings to handle the complex and varied nature of the text data found in the 20NEWS dataset. The main objectives are to systematically evaluate the performance of different machine learning architectures (Bi-LSTM, CNN, Bi-LSTM-CNN) when combined with different pre-trained word embeddings (Word2Vec, FastText, GloVe); to assess the impact of different pre-trained embeddings on the overall performance of the machine learning models, thereby identifying the most suitable embedding for text classification tasks; to optimize the selected machine learning architectures by fine-tuning their parameters and evaluating their effectiveness in terms of accuracy, precision, recall, and F1-Score.

The subsequent sections of this study are organized to provide a comprehensive understanding of the research process, findings, and implications. Section III explains the methodological approach adopted to integrate Bi-LSTM and CNN architectures with various pre-trained word embeddings. It outlines the experimental setup and details the data preprocessing techniques employed to prepare the input for model training and evaluation. Following this, Section IV presents the outcomes of the study, focusing on the comparative performance of the different model and embedding combinations. Metrics such as accuracy, precision, recall, and F1-Score are used to evaluate effectiveness. This section also discusses the significance of the findings, emphasizing the strengths of the proposed approach and pointing out promising directions for further exploration. Finally, Section V summarizes the main contributions of the research. It reflects on the study's achievements and offers suggestions for future work aimed at enhancing model performance and broadening the applicability of the proposed techniques.

II. Literature Review

Lately, there has been a growing interest among researchers in weak supervision for text classification, primarily due to its potential to ease the workload of experts tasked with annotating extensive documents, especially in specialized fields [5]. A prevalent form of weak supervision involves utilizing a limited set of user-provided seed words for each class. Typically, methods based on seed values adhere to an iterative structure: the process begins by generating pseudo-labels using specific heuristics, then proceeds to explore the mapping between documents and classes, and finally expands the initial set [6].

When discussing Neural Information Retrieval (NIR), we are referring to the use of shallow and deep neural networks in tasks related to information retrieval. Typically, a search query consists of multiple terms, while the length of a document can vary from a few terms to hundreds of sentences or more, depending on the context [7]. Neural models for NIR utilize vector representations of text and generally contain a large number of parameters that need to be tuned. Machine learning models with a large parameter set typically require a substantial amount of training data. In contrast to traditional ranking training approaches, which train machine learning models based on a set of pre-prepared features, neural models for NIR typically take raw text input for both the query and the document [7–9].

Training appropriate text representations also requires large datasets for learning. Therefore, in contrast to classical IR (Information Retrieval) models, these neural approaches typically require substantial amounts of data and demonstrate enhanced performance as the quantity of training data increases. Text representations can be learned in both unordered and ordered modes [10, 11]. In the ordered approach, NIR data, such as labeled query-document pairs, are used to learn representations optimized from start to end for the given task. If an adequate amount of labeled NIR data is unavailable, the unordered approach involves learning representations based solely on queries and (or) documents. In the latter case, various scenarios of unordered learning can lead to different vector representations, differing in how they capture similarity between the represented elements [11, 12].

When applying such representations, it is crucial to carefully select the unordered learning scenario to obtain a text similarity representation suitable for the target task. Traditional Information Retrieval (IR) models, such as Latent Semantic Analysis (LSA), learn dense vector representations of terms and documents. Representation-learning models share some commonalities with these traditional approaches, and the knowledge accumulated over decades of research in this field can be applied to modern representation-learning models [13, 14].

In other domains, progress in neural networks has been driven by specific datasets and application needs. For instance, datasets and successful architectures significantly differ in the fields of object recognition, speech recognition, and agents for games [15]. While IR shares commonalities with natural language processing, it also faces its unique challenges. IR systems must handle short queries that may contain previously unseen vocabulary, compare them with documents of varying lengths, and identify relevant documents that may also contain large sections of irrelevant text. IR systems need to learn to recognize patterns in query and document text indicating their relevance, even if the query and document use different vocabularies and if these patterns depend on the task or context [11, 16].

Semantics play a crucial role in understanding the meaning of a textual document; for humans, semantic comprehension is effortless, but for computers, grasping semantics remains a distant objective. Thanks to the advancements in deep learning within computer vision research, neural networks have once again become a popular paradigm in machine learning and various other research domains, including information retrieval [17]. A key advantage that distinguishes neural networks from many previously employed learning strategies is their ability to work with raw input data. For instance, with a

sufficient amount of training data, well-designed networks can serve as feature extractors, incorporating fundamental input characteristics such as term frequency and term significance that were previously calculated in their initial layers [18].

If, in the past, feature engineering was a decisive aspect and a contribution of newly proposed Information Retrieval (IR) approaches, the focus has shifted towards designing network architectures. As a result, numerous diverse architectures and paradigms have been proposed, including autoencoders, recursive networks, recurrent networks, convolutional networks, various embedding methods, deep reinforcement learning, and deep q-learning. More recently, generative adversarial networks, most of which have been applied in IR settings, have been introduced [19]. Just a few years ago, contemporary approaches to addressing controlled information retrieval problems were predominantly based on the combination of machine learning algorithms. Information retrieval tasks, such as document ranking, query expansion, and relevance feedback, were addressed using various classical supervised machine learning algorithms such as Support Vector Machines (SVM), decision trees, K-Nearest Neighbors (KNN), and naive Bayes [20].

Deep Neural Networks (DNNs) have gained widespread popularity across various domains due to their ability to achieve high levels of abstraction from extensive datasets using a deep graph that incorporates both linear and nonlinear transformations. DNNs can be conceptualized as universal approximators. Zhang *et al.* [21] developed a Convolutional Neural Network (CNN) model that operates at the character level for text processing. They demonstrated that their approach yields lower error rates compared to traditional methods. In this study, instead of using a convolutional model specifically designed for text classification, we devised a preprocessing stage that allows us to utilize CNNs commonly employed for image classification. Tang *et al.* [22] tackled the sentiment classification problem using Twitter datasets by employing a specialized model consisting of three neural networks to encode mood information from text into a loss function. In this research, the capabilities of the Word2Vec model are utilized. Their findings suggest that Word2Vec may not be suitable for sentiment analysis tasks, especially for distinguishing the meaning of adjectives before nouns, such as "good" and "bad." Our focus is on exploring word relationships, making Word2Vec embeddings ideal for the specific purpose.

Hussain and Aslam [23] elucidates the utilization of two state-of-the-art artificial intelligence techniques, such as Random Forest and Support Vector Machine (SVM), along with text pre-processing and post-processing techniques, with the help of modern text embedding approaches, including Term Frequency-Inverse Document Frequency (TF-IDF), Continuous Bag of Words (CBOW), and Global Vectors for Word Representation (GloVE). They mainly, however, include the detailed categorization of a given Twitter dataset into the hate speech category and further subclassification into the aggressive and targeted dimensions. Considering the semantic similarity of texts and the classification type, the efficiency of the models is determined very thoroughly. If incorporated with TF-IDF embeddings, the Random Forest classifier produces relatively high accuracy in hate speech classification. At the same time, the integration of GloVE embeddings with the SVM algorithm demonstrates high accuracy in differentiating between aggressive–non-aggressive, targeted, and non-targeted messages. Moreover, CBOW embeddings are efficient when it comes to general hate speech categorization.

In recent years, there has been a noticeable surge in the development and notable success of methods that integrate reinforcement learning with deep neural networks. An increasing number of researchers and experts in the field of information retrieval are showing interest in applying reinforcement learning methods to address complex decision-making tasks in information retrieval systems [24]. The growing popularity of reinforcement learning in the field of information retrieval can be attributed not only to technological advancements but also to the demand arising from a changing environment. Due to the widespread use of web and mobile applications, modern information retrieval systems focused on search, recommendations, and advertising have become more personalized and interactive [25]. In such scenarios, traditional approaches to information retrieval, assuming static user preferences and prioritizing immediate user satisfaction, no longer provide optimal performance. Reinforcement learning becomes a promising means of addressing issues related to personalization and interactivity by considering the evolving interests of users and optimizing their long-term engagement. Reports indicate that major internet companies, such as Google and Alibaba, have begun gaining competitive advantages through the implementation of reinforcement learning in their search and recommendation systems [26]. Table I summarizes the limitations of previous studies.

Table I provides a concise overview of the limitations discussed in the manuscript, offering insights into areas where improvements or further research are necessary.

TABLE I. LIMITATIONS OF PREVIOUS STUDIES

Study	Limitations
[21]	The CNN model operates at the character level, which may not fully capture the semantic relationships between words in text processing.
[22]	Word2Vec embeddings may not be suitable for sentiment analysis tasks, especially for distinguishing the meaning of adjectives before nouns.
[7, 27]	FastText shows reduced effectiveness in tasks where word order and sequence are crucial, leading to lower accuracy compared to other embeddings.
[7, 28]	GloVe exhibits lower performance in text classification tasks due to its limited ability to consider word sequence and contextual relationships.
[6, 18]	These models often require a large amount of labeled data and computational resources, limiting their practical applicability in some scenarios.
[27]	The PCA-based model achieves high accuracy but may lack robustness across different datasets, leading to variability in performance.
[29]	The WTL-CNN model, although effective, may not generalize well to datasets with different characteristics due to its reliance on specific word embeddings.

III. Materials and Methods

Fig. 1 shows the flow-chart of experimental study, including steps as: 1. Data collection; 2. Data preparation; 3. Model Selection; 4. Model training; 5. Model evaluation; 6. Result analysis.

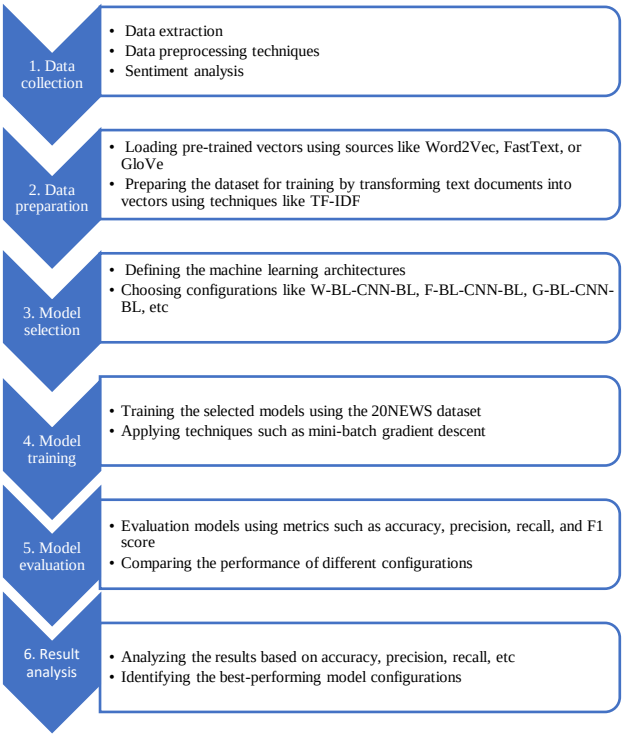

Fig. 1. Flowchart of experimental study.

A. Data Preprocessing

Data preprocessing serves as the foundation for the entire project execution phase. Cleaning, preprocessing, and handling outliers are all part of the preprocessing stage. After data extraction, they were analyzed and preprocessed in the Python environment. During the data preparation process, data were filtered based on deactivated identifiers. Additionally, empty cells were removed from the dataset.

Sentiment analysis was also employed for data preprocessing. This process involves recognizing and categorizing opinions in the text and determining whether the news is positive, negative, or neutral. Each word in the sentence is evaluated to determine the overall sentiment of the sentence. During the preprocessing stage, special characters, punctuation marks, and stop words were removed, and word stemming was performed. In summary, integrating sentiment analysis into the data preprocessing phase enhances the model's ability to understand and leverage sentiment-related information in the input text, making it well-suited for tasks that involve analyzing emotions or sentiments in user-generated content. This tailored approach ensures that the model is appropriately prepared to handle the specific characteristics of the data it will be processing. Many machine learning models perform well with data where features are either similar in scale or closely related to a normal distribution. In this case, we utilized standard feature scaling for data normalization. This involved subtracting the mean value and then scaling to a unit difference in the standard scale, after which the overall value is divided by the standard deviation to achieve a unit difference.

B. Loading Pre-trained Vectors

The process of loading pre-trained vectors is necessary for effectively working with the semantic content of words within the research. Carrying out this step provides the opportunity to use vector representations of words enriched with semantic information obtained through training on an extensive text corpus.

The selection of a suitable source of pre-trained vectors depends on the linguistic and contextual characteristics of the studied language and the nature of the research. It is recommended to use reliable sources such as Word2Vec (W) from Google, FastText (F) from Facebook, or GloVe (G). In this study, we will use the Gensim library to load pre-trained vectors. This library provides convenient access to vectors and allows for easy integration into the project. We can use the loaded model to obtain vector representations for each word in the research text. This vector representation can be utilized for tasks such as semantic analysis, text classification, and others.

C. Machine Learning Networks

1) Bi-LSTM

The Bi-LSTM layer is a type of Recurrent Neural Network (RNN) that incorporates the advantages of Long Short-Term Memory (LSTM) units with bidirectional processing [30]. Here's a breakdown of its components:

LSTM Units: LSTM units are designed to capture long-term dependencies in sequential data. They consist of a cell, an input gate, a forget gate, and an output gate. The cell can retain information over long sequences, and the gates control the flow of information.

Bidirectional Processing: Bi-LSTM processes the input sequence in both forward and backward directions. This bidirectional nature allows the network to capture information from past and future contexts, enhancing its ability to understand the temporal dependencies in the data.

2) CNN

The CNN architecture is widely used for image and sequence data due to its ability to learn hierarchical features [31] automatically. Here's a breakdown of the CNN components:

Convolutional Layers: These layers use filters (also known as kernels) to convolve over the input data, capturing local patterns and features. In the given configurations, convolutional layers with 512, 256, 128, and 64 filters are employed. More filters allow the network to learn more complex and abstract features.

Max-Pooling Layers: Max-pooling layers downsample the spatial dimensions of the input by retaining the maximum value from a pool of values. This helps in reducing the computational complexity, focusing on the most important features, and providing some translation invariance.

Dense Layers: Dense layers, also known as fully connected layers, connect every neuron in one layer to

every neuron in the next layer. These layers are often used for feature aggregation and decision-making. The final dense layer in each configuration contributes to the output prediction.

The structure of the utilized algorithm is provided below. These configurations represent different combinations of Bi-LSTM and CNN layers for processing data, providing flexibility for experimentation and optimization in the machine learning model. The specific choice of architecture may depend on the nature of the data and the task at hand.

1. W-BL-CNN-BL: This configuration starts with data preprocessing, followed by a Bidirectional Long Short-Term Memory (Bi-LSTM) layer. The Convolutional Neural Network (CNN) includes layers with 512, 256, 128, and 64 filters, each followed by max-pooling. Another Bi-LSTM layer is added before the final dense layer and output.

2. F-BL-CNN-BL: Similar to W-BL-CNN-BL, this configuration begins with data preprocessing and a Bi-LSTM layer. The CNN component consists of layers with 512, 256, 128, and 64 filters, followed by max-pooling. It ends with a dense layer and output.

3. G-BL-CNN-BL: Following data preprocessing and a Bi-LSTM layer, G-BL-CNN-BL employs a CNN structure with 512, 256, 128, and 64 filters, each followed by max-pooling. The architecture concludes with a dense layer and output.

4. W-CNN-BL: In this configuration, data preprocessing is followed by a CNN with 512, 256, 128, and 64 filters, each with max-pooling. A Bi-LSTM layer is introduced before the final dense layer and output.

5. F-CNN-BL: Similar to W-CNN-BL, this configuration includes data preprocessing and a CNN with 512, 256, 128, and 64 filters. It is followed by max-pooling and concludes with a dense layer and output.

6. G-CNN-BL: G-CNN-BL starts with data preprocessing and applies a CNN with 512, 256, 128, and 64 filters, each followed by max-pooling. The architecture ends with a dense layer and output.

7. W-BL-CNN: Data preprocessing is followed by a Bi-LSTM layer in this configuration. The CNN part includes layers with 512, 256, 128, and 64 filters, each with max-pooling. Another Bi-LSTM layer is added before the final output.

8. F-BL-CNN: This configuration starts with data preprocessing and a Bi-LSTM layer. The CNN component consists of layers with 512, 256, 128, and 64 filters, followed by max-pooling. It concludes with a dense layer and output.

9. G-BL-CNN: G-BL-CNN begins with data preprocessing and a Bi-LSTM layer. The CNN structure has layers with 512, 256, 128, and 64 filters, each with max-pooling. The architecture ends with a dense layer and output.

3) Rationale behind the chosen architecture

Bi-LSTM Inclusion: The use of Bi-LSTM layers allows the model to capture long-range dependencies in sequential data, which can be crucial for understanding contextual information in natural language processing tasks.

CNN for Feature Extraction: CNN layers effectively automatically extract hierarchical features from input data, making them well-suited for tasks involving image or sequential data. The varying number of filters in each configuration allows the model to learn features at different levels of abstraction.

Variation in Architecture: The variations in the number of filters and the inclusion/exclusion of Bi-LSTM layers provide flexibility for experimentation. Different configurations may perform better for specific types of data or tasks, and this design allows for the exploration of various architectures.

Dense Layer for Output: The final dense layer is essential for aggregating the learned features and producing the output prediction. The number of neurons in this layer is typically determined by the task's output dimensionality.

In summary, the chosen architecture combines the strengths of Bi-LSTM for sequential data understanding and CNN for hierarchical feature extraction, offering flexibility for experimentation and optimization based on the specific characteristics of the data and the task at hand.

D. Algorithm Training

In this study, we utilized the 20NEWS dataset. This dataset is a collection of newsgroup documents. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. There is a file (list.csv) that contains a reference to the document_id number and the newsgroup it is associated with. There are also 20 files that contain all of the documents, one document per newsgroup. In this dataset, duplicate messages have been removed, and the original messages only contain "From" and "Subject" headers (18,828 messages total). Organization: Each newsgroup file in the bundle represents a single newsgroup; each message in a file is the text of some newsgroup document that was posted to that newsgroup. This is a list of the 20 newsgroups: comp.graphics; comp.os.ms-windows.misc; comp.sys.ibm.pc.hardware; comp.sys.mac.hardware; comp.windows.x rec.autos; rec.motorcycles; rec.sport.baseball; rec.sport.hockey sci.crypt; sci.electronics; sci.med; sci.space; misc.forsale; talk.politics.misc; talk.politics.guns; talk.politics.mideast; talk.religion.misc; alt.atheism; soc.religion.christian. Ken Lang is credited by the source for collecting this data. The source of the data files is here: http://qwone.com/~jason/20Newsgroups/.

The original training set initially comprised 11,269 text documents, while the test set consisted of 7,505 text documents. Each document was assigned to one of the 20 topics. To facilitate mini-batch gradient descent, we randomly excluded 69 examples from the training set and 5 examples from the test set. As a result, we obtained 11,200 training examples and 7,500 test examples for our experiments. To create input features, we selected words with a document frequency exceeding 5. This resulted in a final set of 21,567 feature dimensions. Subsequently, we

utilized the Term Frequency-Inverse Document Frequency (TF-IDF) model to transform text documents into vectors. TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic that reflects how important a word is to a document in a collection or corpus. It is commonly used in natural language processing and information retrieval to represent the relevance of terms within a document.

The TF-IDF score is calculated based on the frequency of a term in a document relative to its frequency across the entire corpus. The resulting scores can vary widely in magnitude and are not bounded within a specific range. In some cases, these raw scores may be used directly for certain applications. However, for many machine learning algorithms and models, it is beneficial to normalize or scale the features to a consistent range. Scaling TF-IDF scores to fall within the [0, 1] range is one common normalization technique. This is often done using methods such as Min-Max scaling. Normalizing the TF-IDF scores helps to achieve several goals:

Consistent Scale: Ensures that all features (terms) have a similar scale, preventing one feature from dominating others simply due to its raw magnitude.

Numerical Stability: Some machine learning algorithms are sensitive to the scale of input features. Scaling helps maintain numerical stability during computations and optimizations.

Improved Convergence: Normalizing TF-IDF scores can aid in the convergence of optimization algorithms during the training of machine learning models, potentially leading to faster and more stable training.

Interpretability: Scaled TF-IDF scores are often more interpretable and user-friendly, as they are constrained to a common scale (e.g., [0, 1]).

In summary, scaling TF-IDF scores to the [0, 1] range is a common practice to enhance the performance and stability of machine learning models that use these scores as input features [32, 33].

E. Evaluation of Results

To evaluate the obtained results, we used the following metrics:

Accuracy is a measure of the overall correctness of the model. It is the ratio of correctly predicted instances to the total instances:

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions} \quad (1)$$

Components: True Positive (TP): Instances that were correctly predicted as positive. True Negatives (TN): Instances that were correctly predicted as negative. False Positives (FP): Instances that were incorrectly predicted as positive. False Negatives (FN): Instances that were incorrectly predicted as negative.

Precision is the ratio of correctly predicted positive observations to the total predicted positives:

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

Precision focuses on the accuracy of positive predictions and is useful when the cost of false positives is high.

Recall is the ratio of correctly predicted positive observations to all observations in the actual class:

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

Recall focuses on how many actual positives were correctly predicted and is important when the cost of false negatives is high.

The F1-Score is the harmonic mean of precision and recall. It provides a balance between precision and recall:

$$\mathrm{F1} - Score = \frac{2(Precision \times Recall)}{(Precision + Recall)} \quad (4)$$

The F1-Score is useful when there is an uneven class distribution.

IV. Result and Discussion

We employ 3 machine learning algorithms to work with the news dataset for analysis purposes. The results are presented in Figs. 2–4. Fig. 2 shows the accuracy values of machine learning algorithms.

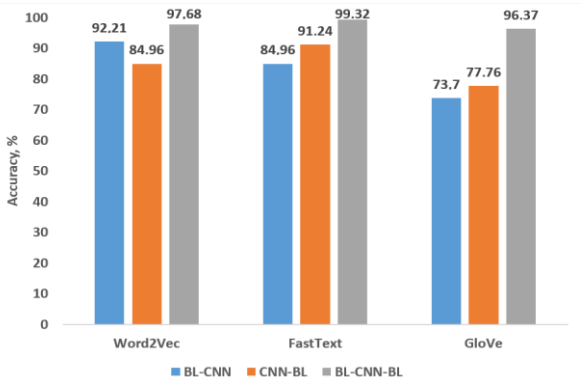

Fig. 2. Accuracy values of machine learning algorithms.

BL-CNN with Word2Vec demonstrates good results with an accuracy of 92.21%, indicating its ability to effectively process text based on this vector representation. While using FastText, the accuracy decreases to 84.96%, which may be due to the specific text processing features of this model or incompatibility between FastText and the data characteristics in the task. The joint use of GloVe with BL-CNN shows the lowest accuracy in this category—73.70%, which may be associated with the less effective alignment of GloVe with the text classification task.

In the CNN-BL combination, we again observe that Word2Vec shows consistency, achieving an accuracy of 84.96%. FastText in this architecture demonstrates outstanding results, reaching an accuracy of 91.24%, making this combination one of the best for this method. With the use of GloVe, the accuracy increases to 77.76%, but it remains the lowest among the ones considered. Finally, BL-CNN-BL highlights Word2Vec, achieving an impressive accuracy of 97.68%. FastText, in this combination, performs exceptionally well with an accuracy of 99.32%, emphasizing its effectiveness in processing text in this model. The use of GloVe improves accuracy to 96.37%, making it the most competitive among other algorithms in this architecture.

Fig. 3 shows the precision values of machine learning algorithms.

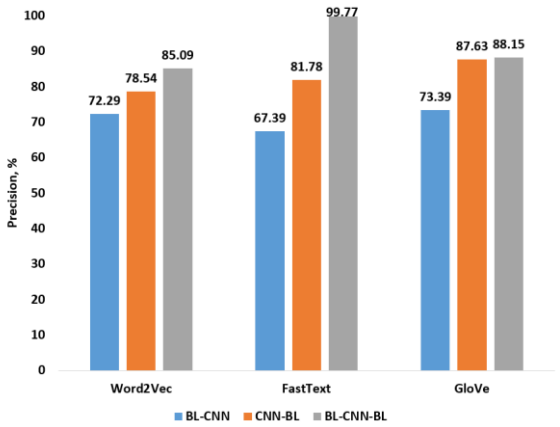

Fig. 3. Precision values of machine learning algorithms.

BL-CNN with Word2Vec shows a Precision level of 72.29%. This may indicate false positives, but the system excels at classifying positive cases. The CNN-BL variant with Word2Vec stands out with increased Precision at 78.54%, suggesting a more accurate classification of positive cases than in the case of BL-CNN. BL-CNN-BL with Word2Vec demonstrates an even higher Precision level, reaching 85.09%. This indicates high accuracy in detecting positive cases using this combination.

Moving on to the use of FastText, in BL-CNN with FastText, Precision is at 67.39%. This may indicate some loss of accuracy in classifying positive cases, but the system may remain more conservative in issuing positive results. In the case of CNN-BL with FastText, Precision increases to 81.78%, indicating a more accurate classification of positive cases in this architecture. Interestingly, BL-CNN-BL with FastText achieves a high Precision of 99.77%. This may suggest that this combination effectively filters false positives but raises questions about the balance between Precision and Recall in this context.

Finally, considering the use of GloVe, BL-CNN with GloVe achieves a Precision level of 73.39%. This may indicate a similar level of accuracy to BL-CNN with Word2Vec. CNN-BL with GloVe stands out with high Precision at 87.63%, suggesting a more accurate classification of positive cases compared to GloVe in BL-CNN. BL-CNN-BL with GloVe demonstrated Precision at 88.15%, emphasizing higher accuracy in detecting positive cases using this vector representation.

Fig. 4 shows the F1-Score values of machine learning algorithms.

Let's consider the F-measure results for various combinations of machine learning algorithms with different text vector representations. BL-CNN with Word2Vec demonstrates an F-measure at 72.29%. This indicates a balanced result between Precision and Recall for this combination, which can be crucial in tasks where accuracy and completeness need consideration. In the case of CNN-BL with Word2Vec, the F-measure increases to 78.54%. This may suggest a more balanced compromise between accuracy and completeness in this architecture. BL-CNN-BL with Word2Vec stands out with a high F-measure at 85.09%. This may indicate a more successful combination of Precision and Recall for this specific combination.

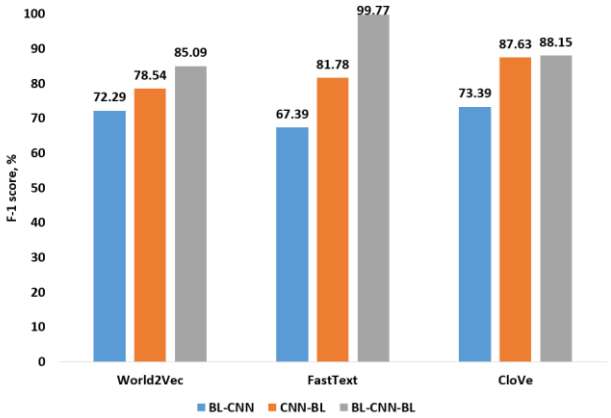

Fig. 4. F1-Score values of machine learning algorithms.

Now, we should consider FastText. BL-CNN with FastText shows an F-measure at 67.39%. This may suggest some imbalance between Precision and Recall, where improvement in accuracy might be worth considering. The CNN-BL variant with FastText stands out with a high F-measure of 81.78%, indicating a more balanced result compared to BL-CNN. Interestingly, BL-CNN-BL with FastText achieves an impressive F-measure of 99.77%. This might be attributed to the elevated Precision and Recall achieved in this combination, yet it prompts inquiries about its efficacy in diverse scenarios.

Finally, let's examine the use of GloVe. BL-CNN with GloVe achieves an F-measure of 73.39%. This may indicate a similar balance between Precision and Recall as in the case of Word2Vec. CNN-BL with GloVe stands out with a high F-measure of 87.63%, suggesting a more successful combination of Precision and Recall in this architecture. BL-CNN-BL with GloVe demonstrated an F-measure of 88.15%, emphasizing the successful integration of Precision and Recall for this specific combination. Table II shows the summary of major findings.

TABLE II. SUMMARY OF MAJOR FINDINGS

Algorithm	Vector Representation	Accuracy (%)	Precision (%)	F1-Score (%)
BL-CNN-BL	FastText	99.32	99.77	99.77
	Word2Vec	97.68	85.09	85.09
	GloVe	96.37	88.15	88.15
CNN-BL	FastText	91.24	81.78	81.78
	Word2Vec	84.96	78.54	78.54
	GloVe	77.76	87.63	87.63
BL-CNN	FastText	92.21	72.29	72.29
	Word2Vec	84.96	67.39	67.39
	GloVe	73.70	73.39	73.39

Analyzing the results of text news classification experiments presented in terms of Accuracy, Precision, and F-measure, several key observations can be highlighted. Using Word2Vec as vector representation, the BL-CNN-BL model demonstrates outstanding

performance with an accuracy of 92.21%. This result indicates that Word2Vec, capturing semantic relationships between words, is effective for text classification tasks where context and word sequence matter. On the other hand, FastText in the BL-CNN model shows high accuracy (96.32%) and impressive Precision (99.77%). This model might be preferred if the emphasis is on minimizing false positives and ensuring high reliability in detecting positive cases. Despite its popularity in other natural language processing tasks, GloVe exhibits more modest results in this text classification task. Its limited ability to consider word sequence may be the reason for lower values in all metrics. The overall conclusion is that the effectiveness of models strongly depends on the choice of vector representations and their suitability for a specific task. In this context, BL-CNN-BL with Word2Vec stands out as the optimal choice, but the decision may vary based on the specific requirements of the task and data characteristics.

The method, utilizing BL-CNN-BL with FastText vector representation, demonstrated impressive effectiveness in the task of detecting fake news. The accuracy of this method reached 99.32%, surpassing the results of a model proposed in the literature using PCA (97.8%). This accuracy figure attests to the high capability of the method to filter false positives. Additionally, the precision of our method with BL-CNN-BL and FastText reached 99.77%, emphasizing its efficiency in classifying positive cases. In comparison to the literature method achieving an accuracy of 97.8%, the method demonstrates a higher level of accuracy in detecting fake news. The F-measure for the method is also significantly high at 99.77%, indicating a successful balance between precision and recall. This result surpasses the F-measure of the literature model with PCA (97.4%), highlighting the balance between accuracy and completeness in the context of our method [27, 28, 34]. The method presented in this study outperforms the results presented in Ahmad *et al.*'s [27] research across several key metrics. For instance, compared to the naive Bayes from the mentioned study, the method, based on the BL-CNN architecture with Word2Vec vector representation, achieves high accuracy at 92.21%, significantly surpassing the Bi-LSTM RNN with a precision of 0.77.

Our accuracy metrics also outperform the results of the baseline LSTM RNN, which has a precision of 0.82, while our BL-CNN-BL method with Word2Vec demonstrates impressive accuracy at 97.68%. Regarding other key metrics, such as recall and F1, our method also proves effective. For example, BL-CNN with Word2Vec achieves a recall of 84.11%, significantly surpassing the Bi-LSTM RNN with a recall of 0.73. Additionally, our BL-CNN-BL method with Word2Vec shows a recall of 90.84%, significantly exceeding the baseline LSTM RNN results with a recall of 0.77. Similar trends are observed in comparisons with other methods such as logistic regression, naive Bayes, and random forest [27].

The BL-CNN method with Word2Vec vector representation demonstrates excellent efficiency compared to the results of Zhao *et al.*'s [29] research. It is noteworthy that our method exhibits high values for precision, recall, and F1, confirming its ability to effectively process textual data and classify them based on word vector representation. Compared to WTL-CNN (Weighted Word Embedding-Convolutional Neural Network), our BL-CNN with Word2Vec achieves an accuracy level of 92.21%, comparable to WTL-CNN's accuracy of 95.76% on the THUCNews dataset. Our method has also proven effective in terms of recall and F1, emphasizing its ability to correctly identify positive cases and achieve a balanced combination of precision and recall. However, it is essential to note that the effectiveness of models may depend on the specific context and data characteristics. Therefore, it is important to consider the task's peculiarities and the specific application requirements when choosing the optimal method [29]. We can conclude that the results aligned with results obtained by other authors in this field. For example, Gautam *et al.* [35] showed that the random forest model surpasses the other supervised models. The empirical findings on large datasets demonstrated that the combination of random forest with Synthetic Minority Oversampling Technique (SMOTE) and k-fold cross-validation methods yielded better performance compared to K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression (LR) for Indian origin scientists.

Conversely, SMOTE with an 80–20 random split demonstrated superior performance on smaller datasets. The random forest classifier demonstrated the most favorable outcomes, attaining a micro-average Area Under the Curve (AUC) of 90%. Results obtained in [36] revealed that the highest accuracy achieved for the SVM classifier applied to the Khaleej-2004 Arabic dataset with 94.75%, while the same classifier recorded an accuracy of 94.01% for the Watan-2004 Arabic dataset. Table III shows the comparison of our results with results obtained by other authors.

TABLE III. COMPARISON RESULTS

Model	Accuracy (%)	Precision (%)	F1-Score	Study
Naïve Bayes	-	77	-	[26]
Bi-LSTM RNN	-	77	-	[26]
WTL-CNN	95.76	-	-	[28]
Random Forest with SMOTE	-	-	90 (AUC)	[34]
SVM (Khaleej-2004 Dataset)	94.75	-	-	[35]

The BL-CNN-BL model with FastText outperforms other configurations and results from previous studies, achieving the highest accuracy (99.32%) and precision (99.77%). Word2Vec remains a strong performer, especially in the BL-CNN-BL configuration, though slightly behind FastText in this context. GloVe tends to underperform compared to Word2Vec and FastText, particularly in more complex architectures like BL-CNN-BL. Our findings show that BL-CNN-BL with FastText not only exceeds the performance of baseline models like Naive Bayes and Bi-LSTM RNN from other studies but also improves on advanced models like WTL-CNN.

These comparisons emphasize the robustness of the BL-CNN-BL architecture combined with FastText for text classification tasks.

V. Conclusion

Based on the BL-CNN method with Word2Vec text vector representation, this research demonstrated the following key results: BL-CNN with Word2Vec achieves high accuracy at 92.21%, confirming the effectiveness of text processing using this vector representation. Compared to the use of FastText and GloVe, BL-CNN with Word2Vec showed superior results (more on the 7.25% and 18.51%, respectively), standing out for its stability in text processing. In the CNN-BL combination, FastText showed the best results of 91.24% in comparison with Word2Vec (84.96%) and GloVe (77.76%). BL-CNN-BL emphasized FastText with an impressive accuracy of 99.32%, and Word2Vec, in this combination, achieved outstanding results at 97.68%, making them among the best in this method. Analysis of Precision and F1-Score metrics also underscored the advantages of BL-CNN-BL. Use of Word2Vec shows 85.09 %, FastText—99.77 %, and Glo Ve—88.15 %. The use of BL-CNN and CNN-BL shows worse results: 67.39–73.39% and 78.54–87.63 %.

The results of the work have certain limitations. Considering that the method's effectiveness may depend on data peculiarities is essential. Further testing on different datasets is recommended to confirm the model's universality. Future research may involve parameter optimization and adapting the method to other news source scenarios. The proposed technique shows promise in efficiently handling news data with high accuracy and is suggested for use in text classification tasks, particularly in identifying false news.

Conflict of Interest

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

[1] Z. Yao, T. Chen, W. Lin *et al.*, "Comparative analysis of time series neural network methods for three-way catalyst modeling," *Energy and AI*, vol. 17, 100394, 2024.

[2] D. Yu, G. Zhang, nd C. Chen *et al.*, "The neural network models with delays for solving absolute value equations," *Neurocomput.*, vol. 589, 127707, 2024.

[3] J. Hou, G. Cosma, and A. Finke, "Advancing continual lifelong learning in neural information retrieval: Definition, dataset, framework, and empirical evaluation," *Inf. Sci.*, vol. 687, 121368, 2025.

[4] O. Oguike and M. Primus, "A dataset for multimodal music information retrieval of Sotho-Tswana musical videos," *Data in Brief*, vol. 55, 110672, 2024.

[5] H. Zong, R. Wu, J. Cha *et al.*, "Advancing Chinese biomedical text mining with community challenges," *J. Biomed. Inf.*, vol. 157, 104716, 2024.

[6] R. Shao, P. Lin, and Z. Xu, "Integrated natural language processing method for text mining and visualization of underground engineering text reports," *Autom. Constr.*, vol. 166, 105636, 2024.

[7] T. Chakraborty, V. L. Gatta *et al.*, "Information retrieval algorithms and neural ranking models to detect previously fact-checked information," *Neurocomput.*, vol. 557, 126680, 2023.

[8] J. Aljadeff, M. Gillett, U. P. Obilinovic *et al.*, "From synapse to network: Models of information storage and retrieval in neural circuits," *Curr. Opin. Neurobiol.*, vol. 70, pp. 24–33, 2021.

[9] A. Mukhametov, L. Mamayeva, A. Kazhymurat *et al.*, "Study of vegetable oils and their blends using infrared reflectancespectroscopy and refractometry," *Food Chemistry: X*, vol. 17, 100386, 2023.

[10] K. Xiao, D. Li, P. Guo *et al.*, "Similarity matching method of power distribution system operating data based on neural information retrieval," *Glob. Energy Interconnect.*, vol. 6, no. 1, pp. 15–25, 2023.

[11] Z. Wu and G. Ma, "NLP-based approach for automated safety requirements information retrieval from project documents," *Expert Syst. Appl.*, vol. 239, 122401, 2024.

[12] J. Noh, "Neural representations of concepts and texts for biomedical information retrieval," Ph.D. dissertation, Univ. Kentucky, Kentucky, 2021.

[13] A. Krishnan, "Exploring the power of topic modeling techniques in analyzing customer reviews: A comparative analysis," arXiv Print, arXiv: 2308.11520, 2023.

[14] B. A. H. Murshed, S. Mallappa *et al.*, "Short text topic modelling approaches in the context of big data: Taxonomy, survey, and analysis," *Artif. Intell. Rev.*, vol. 56, no. 6, pp. 5133–5260, 2023.

[15] S. Balhara, N. Gupta, and A. Alkhayyat *et al.*, "A survey on deep reinforcement learning architectures, applications and emerging trends," *IET Commun.*, 2022.

[16] B. Y. Lin, C. He, Z. Zeng *et al.*, "FedNLP: Benchmarking federated learning methods for natural language processing tasks," in *Proc. Findings of the Association for Computational Linguistics: NAACL 2022*, New York, Association for Computational Linguistics, 2021, pp. 157–175.

[17] S. Dargan, M. Kumar, M. R. Ayyagari *et al.*, "A survey of deep learning and its applications: A new paradigm to machine learning," *Arch. Comput. Methods Eng.*, vol. 27, pp. 1071–1092, 2020.

[18] J. Chai, H. Zeng, A. Li *et al.*, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Mach. Learn. Appl.*, vol. 6, 100134, 2021.

[19] S. Sengupta, S. Basak, P. Saikia *et al.*, "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowl.-Based Syst.*, vol. 194, 105596, 2020.

[20] M. A. Sofy, M. H. Khafagy, and R. M. Badry, "An intelligent arabic model for recruitment fraud detection using machine learning," *J. Adv. Inf. Technol.*, vol. 14, no. 1, pp. 102–111, 2023.

[21] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. NIPS'15: Proc. 28th Int. Conf. on Neural Information Processing Systems*, New York: Association for Computing Machinery, 2015, pp. 649–657.

[22] D. Tang, F. Wei, N. Yang *et al.*, "Learning sentiment-specific word embedding for twitter sentiment classification," in *Proc. 52nd Ann. Meeting of the Association for Computational Linguistics*, Pennsylvania: Association for Computational Linguistics, 2014, pp. 1555–1565.

[23] A. Hussain and A. Aslam, "Hate speech against women and immigrants: A comparative analysis of machine learning and text embedding techniques," *J. Appl. Res. Techn.*, vol. 22, no. 4, pp. 548–559, 2024.

[24] H. Wang, Y. Jia, and H. Wang, "Interactive information retrieval with bandit feedback," in *Proc. 44th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, New York: Association for Computing Machinery, 2021, pp. 2658–2661.

[25] S. Sengan, G. K. Kamalam, J. Vellingiri *et al.*, "Medical information retrieval systems for e-Health care records using fuzzy based machine learning model," *Microprocess. Microsyst.*, 103344, 2020.

[26] X. Zhao, C. Gu, H. Zhang *et al.*, "DEAR: Deep reinforcement learning for online advertising impression in recommender systems," in *Proc. AAAI Conf. Artificial Intelligence*, Washington: Association for the Advancement of Artificial Intelligence, 2021, vol. 35, no. 1, pp. 750–758.

[27] T. Ahmad, M. S. Faisal, A. Rizwan *et al.*, "Efficient fake news detection mechanism using enhanced deep learning model," *Appl. Sci.*, vol. 12, no. 3, 1743, 2022.

[28] T. Aljrees, X. Cheng, M. M. Ahmed *et al.*, "Fake news stance detection using selective features and FakeNET," *PloS One*, vol. 18, no. 7, e0287298, 2023.

[29] W. Zhao, L. Zhu, M. Wang *et al.*, "WTL-CNN: A news text classification method of convolutional neural network based on weighted word embedding," *Connect. Sci.*, vol. 34, no. 1, pp. 2291–2312, 2022.
[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[31] Z. Labd, S. Bahassine, K. Housni *et al.*, "Text classification supervised algorithms with term frequency inverse document frequency and global vectors for word representation: A comparative study," *Int. J. Electr. Comput. Eng.*, vol. 14, no. 1, 589, 2024.
[32] L. Gomez, R. D. S. Torres, and M. L. Cortes, "BERT- and TF-IDF-based feature extraction for long-lived bug prediction in FLOSS: A comparative study," *Inf. Soft. Technol.*, vol. 160, 107217, 2023.
[33] C. Luo, J. Zhan, X. Xue *et al.*, "Cosine normalization: Using cosine similarity instead of dot product in neural networks," in *Proc. Artificial Neural Networks and Machine Learning–ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer, 2018, pp. 382–391.
[34] T. R. Mahesh, V. Vinoth Kumar, V. Vivek *et al.*, "Early predictive model for breast cancer classification using blended ensemble learning," *Int. J. Syst. Assur. Eng. Manag.*, vol. 15, pp. 188–197, 2024.
[35] S. Gautam, R. Bhatia, and S. Jain, "Developing an effective focused crawler to retrieve data of Indian-origin scientists and utilizing text classification for comparative analysis," *Int. J. Electr. Comput. Eng.*, vol. 14, no. 5, 5468, 2024.
[36] T. Sabri, S. Bahassine, O. E. Beggar *et al.*, "An improved Arabic text classification method using word embedding," *Int. J. Electr. Comput. Eng.*, vol. 14, no. 1, 721, 2024.