The Indonesian Journal of Computer Science
www.ijcs.net
Volume 14, Issue 4, August 2025
https://doi.org/10.33022/ijcs.v14i4.4831

Enhanced Fake News Detection with Domain-Specific Word Embeddings: A TorchText-Based Method for News Semantics Representation.

Garidzira Tinashe Crispen[1], Sikhumbuzo Ngwenya[2]
crispengari@gmail.com[1], ngwenya.skura@gmail.com[2]
[1,2] University of Fort Hare

Article Information

Received : 4 Apr 2025
Revised : 4 Jul 2025
Accepted : 19 Jul 2025

Keywords

Fake News Detection (FND), omain-Specific Word Embeddings (DSWE), News Classification, TorchText

Abstract

The prevalence of misinformation in digital media highlights the need for effective fake news detection methods. This paper presents a novel approach that leverages domain-specific word embeddings, trained specifically on news content, to improve the accuracy of fake news classification. Using TorchText, we generated 128-dimensional embeddings, optimized with Bi-LSTM and GRU models, achieving a test accuracy of 93.51% with a margin of error of 0.255. Two models were developed to classify fake news based on news headlines. The first model using pre-trained embeddings achieved a test accuracy of 96.51% with a margin of error of 0.102, and the second model trained without pre-trained embeddings, resulting in slightly worse resulting in a slightly lower accuracy of 96.23% with a loss of 0.104. The comparison highlights the significant impact of domain-specific integration on model performance. This study demonstrates the value of custom integration to improve semantic representation and fake news detection accuracy, providing a powerful tool to combat misinformation.

A. Introduction

Fake news detection has become increasingly important with the growth of digital media, where misinformation can spread rapidly and influence public opinion and behaviour. With social media platforms enabling rapid and widespread dissemination of information, automated tools for reliable fake news detection have become necessary. Traditional detection methods are often inadequate to address the nuances of language and context in news media [1]. Domain-specific word embeddings provide an enhanced approach by capturing semantic relationships specific to the information domain, which improves model performance by making embeddings more contextually relevant for fake news detection [2]. Using deep learning models such as Bi-LSTM with domain-specific embeddings improves contextual understanding, which is essential to manage complex language patterns present in misinformation [3].

Most existing fake news detection systems are designed using general-purpose embeddings, which may overlook specific linguistic patterns common in news content [4]. Such general embeddings may miss subtle nuances or domain-specific contexts, reducing the accuracy of misinformation detection. By generating domain-specific word embeddings that match news semantics, models can be better trained to recognize and classify fake news with higher accuracy, especially in text data [5]. However, the impact of these domain-specific embeddings on model accuracy and reliability in fake news classification tasks remains to be explored and quantified.

The prevalence of misinformation in online media has led to extensive research on fake news detection, with various approaches emerging in recent years. Traditional machine learning algorithms such as support vector machines Logistic Regression, Naïve Bayes, SVM, Random Forest, and Decision Tree can be used for fake news classification [6]. These models typically rely on handcrafted features such as language patterns, word usage, and sentiment to detect misinformation. Hoti et al. uses traditional algorithms to classify fake news from 100 Albanian news articles, found that Naïve Bayes was the most accurate with 66%, followed by Decision Tree with 60% and SVM with 53%.

Study by [7] uses machine learning sentiment analysis on Dawn news headlines to assess the performance of Pakistani governments, leveraging SVM and sentiment intensity models to assess public sentiment across political regimes over the past decade. Deep learning models, particularly convolutional neural networks (CNNs) and long short-term memory networks (LSTMs), have demonstrated significant advantages in analysing contextual and sequential textual information [8]. CNNs are good at extracting local patterns and hierarchical representations, while LSTMs excel at maintaining long-term dependencies, making both suitable for tasks such as text classification and sentiment analysis [8]. [9] on their study has highlighted that combining these architectures can further improve contextual understanding, as demonstrated in studies comparing CNNs, LSTMs, and other recurrent structures on different datasets, where each model has its strengths based on the complexity and structure of the data.

A study by [10] used a Transformer-based model that demonstrates strong capabilities in understanding complex contexts and word dependencies, making it particularly well-suited for tasks such as fake news detection. Their study showed

that BART bidirectional processing enables a deeper understanding of language, which is essential for accurately identifying misinformation. Similarly, a study by [11] explored the use of Transformer models in fake news classification using BERT, highlighting their effectiveness in capturing nuanced contextual relationships by attaining test accuracy of 99.20%.

Study on multimodal fake news detection has shown that combining text data with images, audio, or video can significantly improve model reliability and accuracy. For example, studies such as [12] illustrate the effectiveness of using text and image features through an event-adversarial neural network, which is designed to integrate different types of data, thereby improving detection accuracy. Additionally, [13] provide a comprehensive review of fake news detection methods, highlighting how multimodal approaches can capture richer context to distinguish real from fake news. These advances in transformer-based and multimodal methods align with our research focus on integrating rich models that enhance semantic and contextual understanding to improve fake news classification.

Word embeddings are important in NLP because they convert text into numerical vectors that preserve semantic relationships between words. Models such as Word2Vec [14] and GloVe [15] have revolutionized this by capturing the syntactic and semantic structure of words. Word2Vec uses a continuous bag of words (CBOW) or skip-word model, while GloVe applies matrix factorization methods based on word co-occurrence. However, more recent approaches such as augmented embeddings using random permutations (EARP) have advanced the field by incorporating word order information, thereby improving the quality of word embeddings [16]. These improvements have been shown to be effective for tasks such as similarity search, demonstrating that including word order can further refine word vector models and their accuracy in NLP applications.

Recent advancements include contextual embeddings, such as BERT and ELMo, which dynamically adjust word representations based on context [17]. These models consider entire sentences when creating embeddings, allowing them to differentiate between words with multiple meanings. Contextual embeddings have demonstrated improved performance across many NLP applications, but they can be computationally demanding and may still benefit from further domain-specific tuning when applied to specialized tasks like fake news detection [18]

Studies have shown that domain-specific integration can significantly improve performance in specialized domains. For example, a study by [19] demonstrated that BioBERT, fine-tuned on biomedical texts, outperformed general BERT models on biomedical text mining tasks. Similarly, a study by [20] found that SciBERT, a model trained on scientific literature, outperformed BERT on tasks requiring the understanding of specialized scientific terminology. In the financial domain, [21] reported that FinBERT, a model trained on financial text, improved the accuracy of financial sentiment analysis by effectively capturing financial terminology and industry-specific context. These results suggest that domain-specific integration can improve the accuracy of NLP models in specialized domains.

In fake news detection, domain-specific embeddings for the news media domain enable models to capture subtleties in language that may indicate deceptive content, such as emotionally charged or polarizing language [10]. Study by [3] have found that using news-specific embeddings enhances the model's sensitivity to

detecting patterns associated with misinformation, thereby improving classification accuracy. This work builds on these findings by developing and evaluating news-specific embeddings, further demonstrating the value of specialized embeddings in the context of fake news classification.

The current literature has not explored the possibility of detecting fake news by training domain-specific embeddings using the TorchText approach. This study aims to fill this gap by developing a model to train 128-dimensional domain-specific news embedding vectors, which will be stored in a text file. Two models will then be trained: one without pre-trained word embeddings and the other using the recorded domain-specific embeddings to detect fake news based on news headlines or titles. Using the news category dataset, this research creates 128-dimensional embeddings that capture news-specific context. This research thus contributes to advancing fake news detection by showcasing how domain-specific embeddings can improve model performance, providing valuable insights for NLP applications in misinformation detection.

B. Research Method

The CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology is widely recognized for structuring data mining projects. It consists of six main phases: business understanding, data understanding, data preparation, modeling, evaluation, and implementation [22]. This methodology helps guide the data mining process, providing a clear structure that allows for better project management and more efficient resource allocation.

The researchers used two Kaggle datasets for this study. The first dataset, titled News Article Classification Dataset for NLP and ML by [23] was used to train domain-specific word embeddings. This dataset consists of five files with news headlines and content from categories such as sports, technology, education, entertainment, and business. The second dataset, titled Fake News Detection by [24], contains two CSV files labeled “fake” and “true,” each providing news headlines and content specific to the respective category.

The news article classification dataset contains five files, and we created five separate pandas DataFrames, one for each file. Each file contains 2,000 examples, with columns for the news title, description, body, URL, and appropriate category label. Table 1 shows the distribution of news categories across the entire dataset.

Table 1. News Category Dataset Distribution

Category	Examples
Sports	2 000
Entertainment	2 000
Technology	2 000
Education	2 000
Business	2 000
Total	**10 000**

The dataset contains balanced examples across the category columns, with a total of 10,000 examples. Each of the five classes has an equal number of 2,000 samples as shown in Table 1. There was no need for class balancing in this which is a process. In imbalanced datasets, accuracy can be misleading as it is often biased

towards the majority class [25]. Therefore, alternative metrics such as precision, recall, and F1 score are more reliable in evaluating model performance [26].

The five data frames were merged into one large dataset for feature extraction. Three main columns (title, description, and content) were selected as features for each row. To expand the dataset, triples were created by associating each title with its corresponding label, content with its label, and description with its corresponding label. This process tripled the size of the dataset, which was then divided into training, testing, and validation sets. The final dataset contained 30 000 instances of matched text, each labelled with its corresponding news category. This process of increasing the dataset was essential because our dataset was initially small, so for that will require many data argumentation techniques [27].

We split the dataset into three subsets: training, testing, and validation. The testing set is allocated 20% of the total data, while the remaining 80% of the data is for training. From this training data, 20% is allocated to the validation set. This approach is commonly used in machine learning workflows to ensure that the model has enough data for training while maintaining separate datasets for evaluation and tuning [28]. Table 2 shows the total number of examples in each subset after splitting.

Table 2. Examples in sets after splitting the News Category Dataset

Set	Examples
Training	19 200
Testing	6 000
Validation	4 800

From Table 2, we can see that our dataset has enough data to train, based on 19 200 examples that are in the training set our model can learn from this without having problem with data deficiency. Figure 1 shows a pie chat plot for the distribution of labels within each set after we have split our data into 3 subsets.

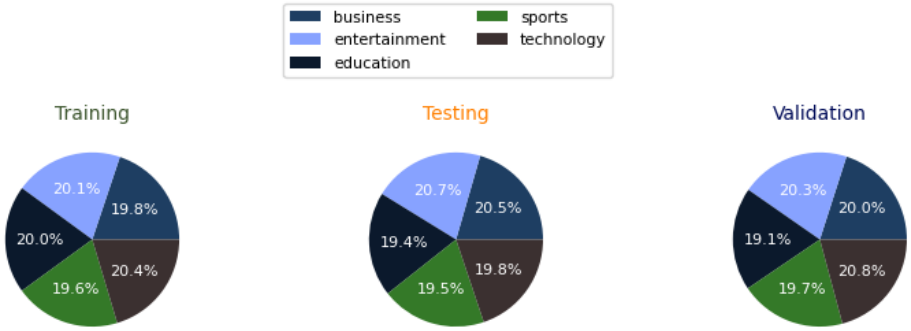


Figure 1. Labels Distributions after data splits.

Figure 1 shows that the labels are evenly distributed across all classes and sets. After spliting the data, we look at the most frequent words in each subset, identifying terms that appear frequently in the text. Table 3 presents word clouds of these common words in the news dataset, providing a clear picture of the important terms that highlight the overall topics and trends in the data.

Table 3. Most common words in each set.

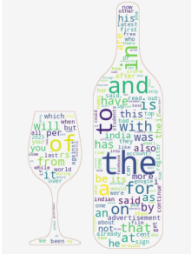
Training	Testing	Validation
		

Table 3 reveals that common words in the dataset include stopwords like “the” and “to.” Although these are frequently removed in typical text preprocessing steps, retaining stopwords is often beneficial for training word embeddings, as they help capture relationships between words in context, which enhances embedding quality. Studies have shown that such syntactic relationships add valuable context and improve downstream NLP task performance [14].

The Fake News Detection dataset consists of two CSV files: one for real news and one for fake news, each containing columns for title, text, subject, and date. For this study, only the title column was extracted from each file and labels were assigned based on the file name (*true.csv* and *fake.csv*). These labelled data entries were then merged into a single complete data frame, consisting of news headlines (headlines) and their corresponding labels. Figure 2 provides a pie chart visualization, illustrating the distribution of labels in the dataset. This approach follows common practices in NLP to simplify dataset labels by file or origin for binary classification tasks [29], [30].

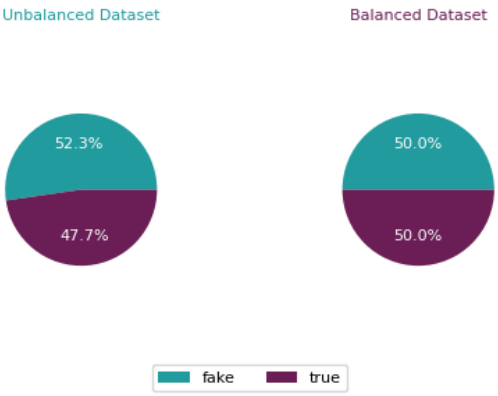


Figure 2. Distribution of labels in the Fake News Detection Dataset.

Figure 2 shows that the fake news class has a larger number of examples than the real news class. To address this imbalance, we applied a down sampling method to fit the class to a minority of examples, ensuring a balanced dataset. Following a similar approach as with the News category dataset, the balanced dataset was then divided into three subsets: training, validation, and testing. Figure 3 presents a pie chart showing the proportion of examples in each subset, which helps visualize the even distribution across classes for training and evaluating model accuracy [25], [26].

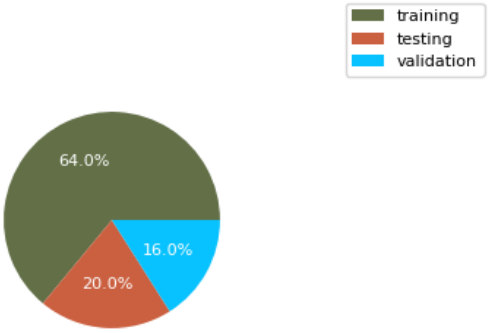


Figure 3. Examples in sets after data splits.

As shown in Figure 3, the dataset is divided into three sets, with 64% allocated to the training set, 20% allocated to testing, and 16% reserved for validation purposes. Table 4 provides details on the number of examples included in each subset, illustrating the distribution between training, testing, and validation for optimal model training and evaluation.

Table 4. Distribution of examples in the Fake News Detection Dataset.

Set	Examples
Training	27 413
Testing	8 567
Validation	6 854
Total	42 834

Table 4 presents the number of examples in each subset of the data, showing that 27 413 of the 42 834 examples were in the training set, 8 567 in the test set, and nearly 6 854 in the validation set. Additionally, Table 5 provides a word cloud visualization of the most common words found in the Fake News Detection dataset for each set, providing an overview of the terms that were frequently used during the training and validation data. This visualization focuses on word usage patterns, which aid in understanding the linguistic structure in the dataset.

Table 5. Most common words in each set.

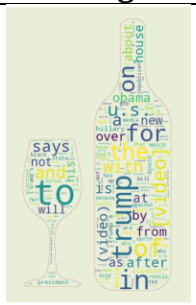
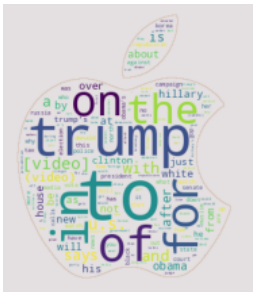
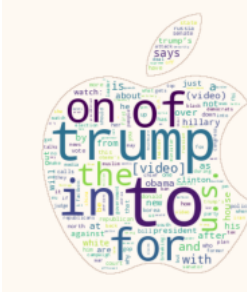
Training	Testing	Validation
		

Table 5 shows that “Trump” is the most frequently occurring word in news headlines, reflecting the prominence of this public figure in the media. This observation confirms the relevance of the dataset to real-world news as it captures

typical trends in contemporary news cycles, confirming that the dataset represents actual news headlines.

The data processing and preparation steps for the News Category and Fake News Detection datasets have similar methods. During data preparation, text features were cleaned by text normalization, a process that involves removing spaces, URLs, numbers, and other textual noise to create cleaner inputs for model training. Tokenization, the step of dividing sentences into a list of words, was performed using spaCy's English tokenization model, following natural language processing (NLP) best practices for managing English corpuses [31]. The vocabulary was created separately from the training dataset, as we recommend using only training data to avoid data leakage, ensuring that the word index mapping remains unbiased. This vocabulary makes it easier to map words to signals, a fundamental step for using the embedding layer during model training and inference. Additionally, this vocabulary is stored to support model prediction and integration generation.

To prepare the labels for training, we converted the text labels to numeric format by assigning each unique label a corresponding index. A dictionary was then created to map each label to its index, an essential step in organizing categorical data for supervised learning. For batch processing, we set the size to 64 and used the *DataLoader* class from PyTorch's *torch.utils.data* module, which allows for efficient data management and improved computational resource management [32]. Data shuffling was enabled only for the training set to improve model robustness, while shuffling was disabled for the validation and test sets to maintain consistency in evaluation [33]

To train embeddings on the News Categories dataset, the model architecture consists of three main layers: an Embedding layer, followed by a Bi-GRU, a Bi-LSTM, and a final linear layer. The model uses backpropagation to optimize learning, training with a text sequence of 128 tokens. The text input is first processed through an embedding layer to generate dense vector representations, which are then fed into a Bi-GRU to capture sequential dependencies in the forward and backward directions. The Bi-LSTM then refines these representations, capturing contextual nuances for better semantic understanding. To mitigate overfitting, we applied a 50% dropout rate across layers, a common technique to improve model generalization and avoid over-reliance on specific neural activations [8].

To further process the output of the Bi-LSTM model, we used a sequence of three linear layers in a sequential configuration to map the transformed embeddings to the final five output layers. Applying a dropout probability of 25% on these linear layers helped avoid overfitting by reducing the model's dependence on specific neurons during training [8]. The model was initialized with the Adam optimizer, which is well suited for tasks involving sparse gradients and large datasets [34] and the *CrossEntropyLoss* function, which is suitable for multi-class classification. The entire model and optimizer were deployed on CUDA-enabled GPUs via Google Colab, which significantly optimized training efficiency and reduced computation time [35]. The number of architectural parameters, as shown in Table 6, represents the model's ability to handle complex word embeddings in this configuration.

Table 6. Bi-RNN Model Parameters

Model	Total Parameters	Trainable Parameters	Non-Trainable Paramers
Bi-RNN	8 613 893	8 613 893	0

Table 6 details the parameters of the Bi-RNN model used to train domain-specific word embeddings on the News Categories dataset. With approximately 9 million trainable parameters, the model was structured for optimal learning with no parameters set as untrainable, thereby maximizing its adaptability during training. The model underwent 20 training epochs, with early stopping due to validation loss to avoid overfitting. The best performing model, determined by the lowest validation loss across multiple epochs, was saved in *.pt* format to ensure reproducibility and ease of deployment. Key training metrics were recorded to allow for detailed evaluation and analysis prior to embedding extraction, which is essential for measuring model performance and understanding embedding quality in subsequent tasks [34]

After training, the best performing model is loaded for evaluation on the test dataset, ensuring its generalizability and reliability. The model's embedding layer, namely with 128-dimensional embedding vectors, was then extracted. Using the pre-recorded vocabulary, each word is associated with its corresponding 128-dimensional vector, thus forming a complete embedding set that matches the dataset. During this extraction, special tokens such as *[pad], [sos], [eos]* and *[unk]* were excluded because these tokens are usually placeholders rather than representative linguistic features. Regarding naming, we adopted similar conventions as GloVe, labeling the saved embedding file as *embeddings.26k.128d.txt*, meaning it contains approximately 26 000 unique words mapped to a 128-dimensional vector, in line with best practices for integrated formatting and interpretability [14], [15].

Two models with identical architectures using Bi-LSTM layers were created. Each model consisted of an embedding layer with a 128-dimensional output, followed by a bidirectional LSTM layer with a dropout probability of 0.5 to avoid overfitting [8]. The final layer was a linear layer that produced a single value, suitable for binary classification tasks. After initializing both models, the total number of parameters for each model was calculated and recorded. Table 7 presents these parameter counts, providing insight into the model complexity and resource requirements.

Table 7. Fake news classification models' parameters.

Model	Total Parameters	Trainable Parameters	Non-Trainable Paramers
With-embeddings	3 206 017	3 206 017	0
Without-embeddings	3 206 017	3 206 017	0

Table 7 shows that all models in the study had an equal number of trainable and non-trainable parameters, with a total of approximately 3 million trainable

parameters. This indicates that there were no non-trainable parameters in the models. The pre-trained domain-specific word embeddings, stored in the *embeddings.26k.128d.txt* file, were loaded to generate a matching embedding matrix for the dataset. This matrix was used in one of the models (the "with embeddings" model), while the other model was run without pre-trained embeddings to provide a comparison point. Such approaches are common in machine learning, as the use of pre-trained embeddings often improves model performance by introducing semantic knowledge [17].

Due to the larger number of parameters in the model, the computational load required the use of GPUs on Google Colab to train the model. Each model was assigned a separate optimizer and loss function. For the loss function, *BCEWithLogitsLoss* was used, which is suitable for binary classification tasks, while the Adam optimizer [34] with default parameters was used to efficiently update the model weights. This combination of tools is widely used for deep learning models because it supports faster convergence and better gradient handling [34].

All models were trained for 10 epochs, with the model with the lowest validation loss during training being recorded as the best performing model for both scenarios. This training method is standard in machine learning to ensure that the model with the best generalization ability is selected, as monitoring the validation loss helps to minimize overfitting [34]. During training, metrics such as accuracy, loss, and other performance indicators were carefully monitored so that the models could be evaluated and compared.

After training, the models were loaded from their respective .pt files and evaluated on the testing dataset, which is essential to evaluate the model performance on unseen data [36]. Evaluation metrics include model classification accuracy, loss, confusion matrix, and classification rate, providing a comprehensive view of model performance, helping to identify strengths and weaknesses [37].

The researchers designed three separate models. The first model aimed to train domain-specific word embeddings in news using a Bi-RNN architecture. This architecture consists of several layers: a convolutional layer, a bidirectional GRU layer, an LSTM layer, and a sequential linear layer. These layers are followed by a final output linear layer with an output dimension of 5, to classify news content into five different categories: sports, education, technology, entertainment, and business. Such an approach allows the model to capture contextual information and sequential dependencies, with the bidirectional GRU and LSTM layers allowing the system to process inputs in both directions, thereby improving semantic understanding [33], [36].

This approach leverages embeddings specifically trained on news corpora to better capture the nuances associated with these categories, following strategies like those observed in previous studies, where custom embeddings are used to improve classification accuracy by tailoring the embeddings to domain-specific terminology and structure [38]. By integrating these components, the architecture aims to improve classification capabilities by combining both GRUs and LSTMs, which have been shown to provide richer feature representations for sequential data tasks [33].

The model was trained to classify five classes using the backpropagation algorithm, a commonly used technique in neural network training that allows for

weights to be adjusted to minimize errors between classes [36]. After training is complete, the best performing model is loaded, with only the embedding layer retained for later use, while the other layers are discarded. This selective loading allows the extraction of weights from the embedding layer, which serve as representations of the features from the News Category dataset [15], [39]. The extracted weights are stored as embedding matrices in a text file, named *embeddings.26k.128d.txt*, where "26k" denotes a vocabulary size of approximately 26 000 words and "128d" denotes a vector representation in 128 dimensions convolved using the layer.

This process follows established practices for generating and storing domain-specific embeddings, as demonstrated in various studies that emphasize the importance of embedding size and vocabulary coverage in capturing sufficient semantic information for subsequent tasks [17].These embeddings can be reused in related tasks because they encapsulate contextual relationships between words in a specific domain, thus contributing to improved model efficiency and accuracy [36].

Figure 4 shows a visual representation of how the embedding vectors were extracted from a model that learned to classify news categories by ignoring other layers after the model has trained and how these embedding weights were loaded from a text file to be used in one of the Bi-LSTM that does fake news detection using the Fake News Detection Dataset.

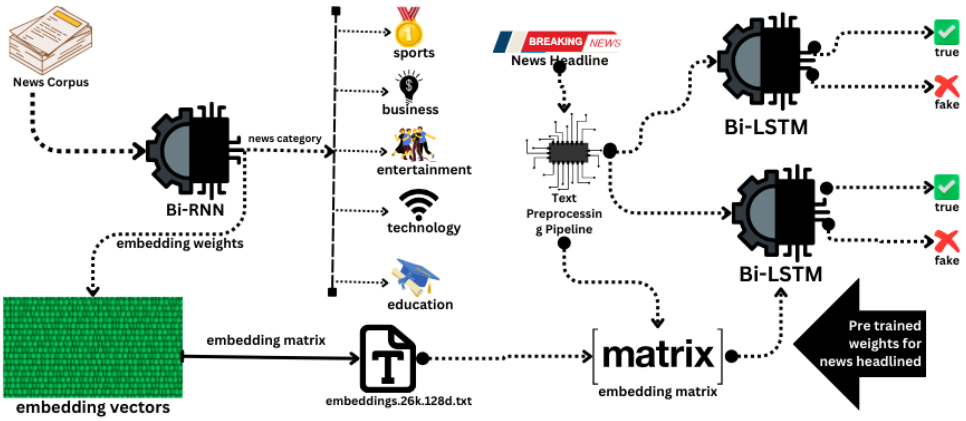


Figure 4. Full Architecture.

After training, the embedding weights are saved to a text file, allowing the new model to classify news headlines as true or fake. Two models with identical Bi-LSTM architectures were developed: one that mines domain-specific word embeddings and the other without pre-trained embedding weights as shown by Fig4. These embeddings are loaded from the text file into a tensor, which produces an embedding matrix that matches the vocabulary of the news dataset [17].

This matrix is then applied to the embedding layer of the first model to improve its ability to understand domain-specific vocabulary. Both models undergo training, evaluation, and testing to predict the authenticity of news headlines, providing comparative insights into the impact of domain-specific embeddings on

performance [39]. The Bi-RNN model is the model that is responsible for training word embeddings using the News Category dataset. Figure 5 shows the whole model architecture of the Bi-RNN model.

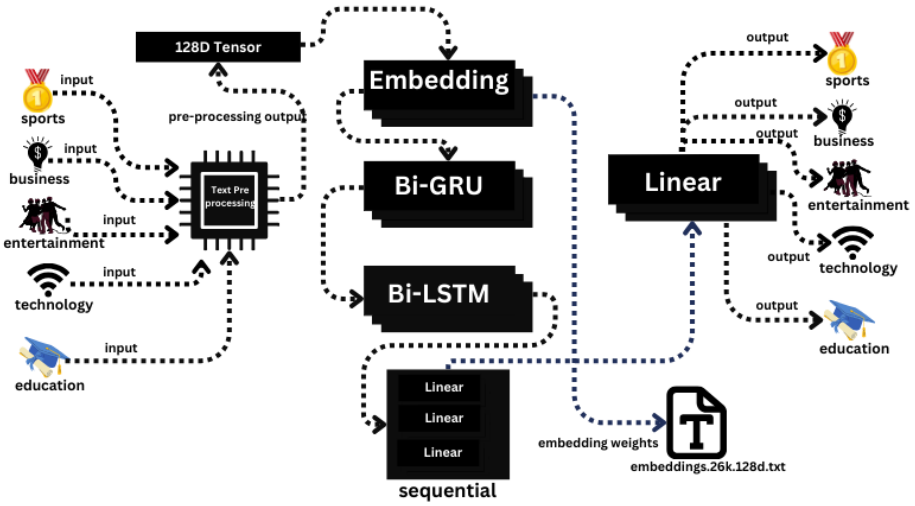


Figure 5. Bi-RNN Model Achitecture.

Figure 5 illustrates the full architecture of the Bi-RNN model, which processes input text such as news headlines, content, or descriptions, classifying them into one of five categories: education, business, technology, sports, or entertainment. The model uses a 128-dimensional text representation that captures complex linguistic patterns to accurately classify news types. By leveraging both GRU and LSTM layers, the model effectively captures contextual nuances across sequences, which is essential for accurate classification in domains with different linguistic styles [40].

The input data in batches, with a batch size of 64, is processed through a text normalization process, which involves removing irrelevant characters and segmenting the text into strings. This preprocessing process produces 128-dimensional tensors, where longer strings are truncated, and shorter strings are padded with a special padding token to maintain consistent input dimensions. The normalized and segmented output is then passed through an embedding layer, which also accepts 128-dimensional input, converting the tokens into dense vector representations [17]. The embeddings are then fed into a Bi-GRU layer, which collects forward and backward contextual information, and finally into a Bi-LSTM layer for advanced sequence processing, allowing for richer feature extraction, required for subsequent tasks [39]

A series of linear layers process the output hidden states of the Bi-LSTM layer, refining them into a 128-dimensional tensor that captures the relevant features learned from the input sequence [36]. This tensor is then fed to the final output linear layer that maps it into five dimensions, corresponding to the model's classification task of distinguishing five news categories (e.g., education, business, technology, sports, and entertainment) [41].

Once trained, the model's embedding layer, which is responsible for converting tokens into meaningful vector representations, retains its learned weights, which

are stored as embeddings in a text file for future applications, as maintaining embeddings helps generalize knowledge and adapt to related tasks [15].

Figure 6 shows the architecture of the Bi-LSTM model which is the model responsible for fake news classification using the Fake News Detection dataset.

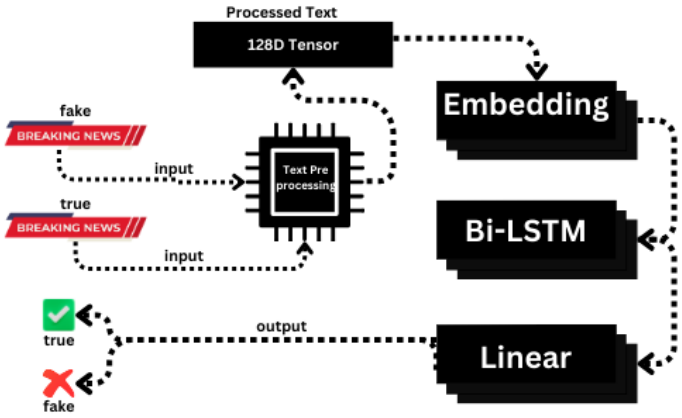


Figure 6: Bi-LSTM model Achitecture.

The text input to determine whether a news story is fake or true is processed through a standardized text preprocessing procedure like that used by the Bi-RNN model, which includes text normalization, encoding, and dimensionality adjustment [17]. The output of this preprocessing stage is a 128-dimensional tensor that undergoes a convolutional transformation. This embedding layer can incorporate domain-specific pre-trained word embeddings, improving the model's ability to recognize patterns in news text related to authenticity [15]. The processed tensor, with a batch size of 64, then passes through a bidirectional LSTM layer, which captures dependencies in both forward and backward text sequences, a technique that has been shown to improve performance in binary classification tasks [21]. Finally, the output is passed to a linear layer configured with a single output dimension, which determines the classification output, determining whether the text is real or fake [41].

C. Result and Discussion

Two different model architectures, Bi-RNN and Bi-LSTM, were developed and trained for this study, with results recorded for each model. Bi-RNN was initially used to train and store domain-specific word embeddings, a technique that has been shown to improve downstream model accuracy in specific domains [17], [42]. Two Bi-LSTM models with identical architectures were then trained to classify news headlines, with one of these models using the pre-trained embeddings obtained from Bi-RNN and the other not incorporating these embeddings. Comparing the two Bi-LSTM models allows for an assessment of the impact of domain-specific embeddings on classification performance. Results from the Bi-RNN, which was used to generate word embeddings, and the two Bi-LSTM models presented below focus on the effectiveness of each architecture in accurately classifying news content. The Bi-RNN model was trained for 20 epochs, and the training history was

stored per epoch for further analysis. After training the model for 20 iterations the following were the outcome in terms of the total training time and last saved epoch.

Table 8. Total training time of Bi-LSTM.

	Value
Total Epochs	20
Testing	5
Total Training Time	13min 23.34 sec

Table 8 shows the model's training history in terms of the total training time and last saved epoch for the best model. The best model was last saved at epoch 5 out of 20 epochs that the model trains for and it took about 13.5 minutes to train the model for 20 iterations. Figure 7 shows how long the model took to complete a single training epoch.

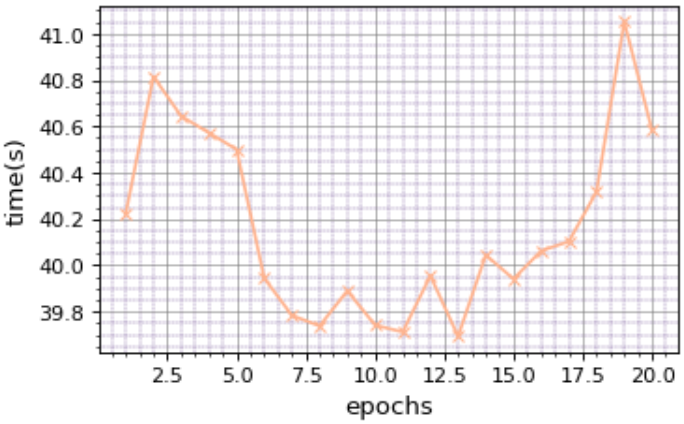


Figure 7. Epoch training time for the Bi-RNN model.

From Figure 7, the model's training time per epoch was not stable. Between epoch 6 and 11 the model's total training time to complete an iteration on average was bellow 40 seconds. Epoch 19 took more training time to complete than the rest of the epochs. Figure 8 shows the training and validation losses that was observed after training the model for 20 epochs.

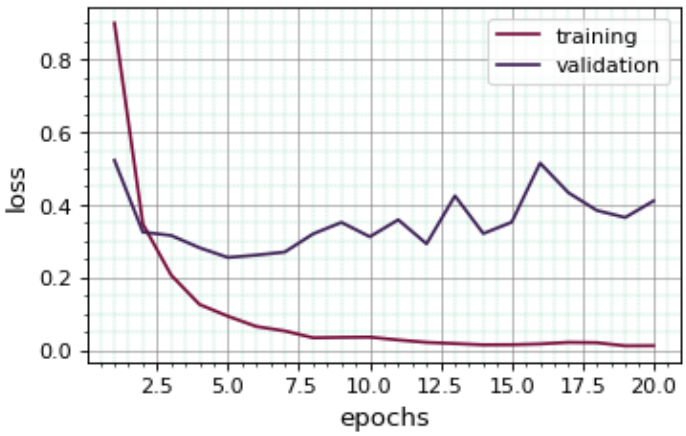


Figure 8. Training and validation epoch loss for the Bi-RNN model.

In the first epoch, the training loss of the model was above 0.8 and the validation loss was 2 units bellow the training loss as shown in Figure 8. The model's training loss gradually fell from epoch 2 all the way to the last epoch, while the validation loss was fluctuating, with a low validation loss being observed at epoch 5. Figure 9 shows the model's training and validation accuracies per epoch that was observed during model training.

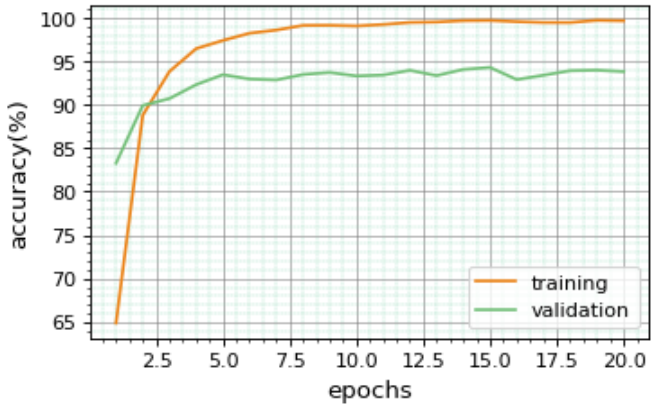


Figure 9. Training and validation accuracies per epoch for the Bi-RNN model.

Figure 9 shows that the model's training accuracy increases gradually from epoch 2 till epoch 15, where the accuracy was just a fraction away to be 100%. On the side of the validation accuracy the accuracy started at bellow 85% and increased to be between 90% and 95% for the whole training period. Table 9 shows the loss and accuracy of the best saved model which was saved on epoch 5 after it has been evaluated on the test dataset.

Table 9. Test loss and accuracy of the best Bi-RNN model.

Metric	Value
Loss	0.255
Accuracy	93.51%

The best saved model had a loss of 0.255 and a test accuracy of 93.51% as illustrated by Table 9. The Bi-RNN performed pretty much well based on these metrics. Figure 10 shows a confusion matrix of the Bi-RNN model after it has been evaluated with the test dataset.

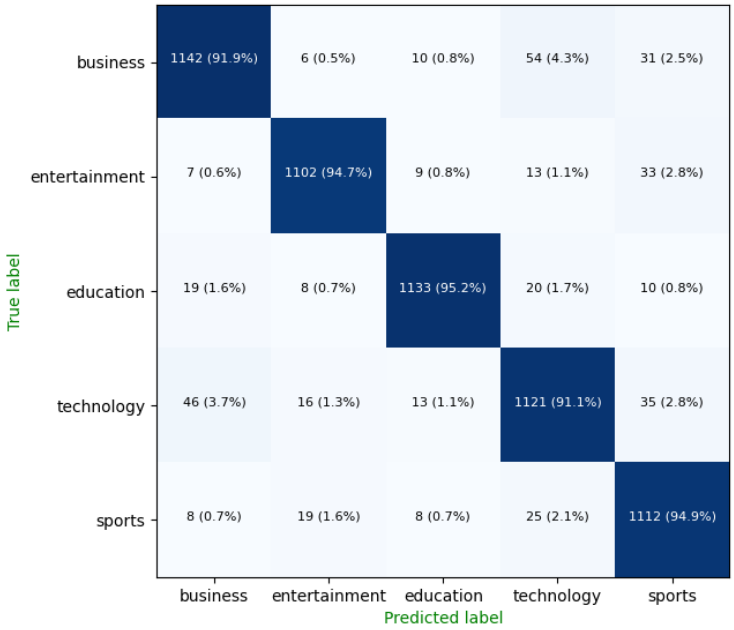


Figure 10. Bi-RNN's confusion matrix.

Figure 10 shows how well the Bi-RNN model classifies each news category. The diagonal values show high accuracy in each class, meaning that the majority of "business," "entertainment," "education," "technology," and "sports" articles are correctly classified. For example, the model correctly identifies 91.5% of "business" articles as "business." Off-diagonal values indicate some misclassification, such as a small percentage of "business" articles labelled as "entertainment" or "technology," which suggests minor overlaps in content between these categories. Figure 11 shows the classification report of the Bi-RNN model after it has been evaluated using the test dataset.

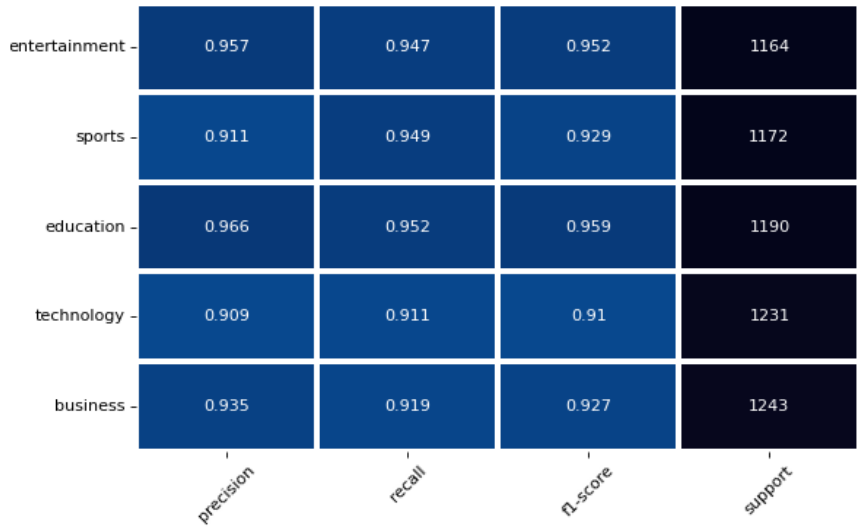


Figure 11. Bi-RNN's classification report.

From Figure 11, demonstrates how strong the model was in identifying different classes using metrics such as precision, recall, and F1-score, which together. High precision scores indicate that the model minimizes false positives, while high recall

scores show that it captures most true positives for each category. The F1-scores above 0.91 across all categories confirm that the model is balanced and effective in classifying news articles accurately. The overall performance metrics underscore the model's robustness in differentiating between categories, though minor improvements could address the slight misclassifications seen in the confusion matrix (Figure 10).

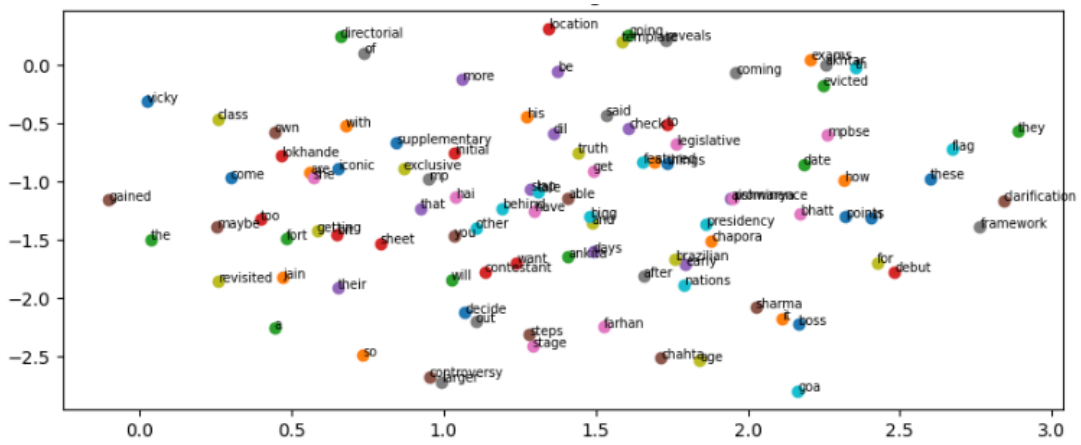


Figure 12. 2-dimensional word embedding visualization.

Figure 12 shows a two-dimensional visualization of the word embedding matrix for the first 100 words, illustrating the relationships between words based on their contextual similarity. Embedding captures semantic proximity; words used in similar contexts will appear closer together in the vector space [14]. For example, “director” and “of” are placed close together, indicating their frequent occurrence in similar contexts. Studies demonstrate that such embedding visualizations reveal semantic clusters, providing valuable information about contextual word usage [14], [15]. Techniques such as t-SNE and PCA often improve the interpretability of these integrations, revealing underlying patterns in the data [17], [35]. Domain-specific embeddings further strengthen these models, mapping words to specialized contexts, such as in fake news detection [43].

The first Bi-LSTM model incorporated pre-trained domain-specific embedding weights into its embedding layer, while the second Bi-LSTM model used randomly initialized weights without applying pre-trained embedding weights. The two models were trained independently, and their training histories were recorded. Table 10 provides details of the total training time and the last recorded epoch for each Bi-LSTM model.

Table 10. Bi-LSTM models' total epoch training time and last saved epoch.

Model Name	Total Epochs	Last Saved Epoch	Total Traing Time
With pretrained embedding weights.	10	4	4min 46.99sec
Without pretrained word embeddings	10	2	4min 47.36sec

The model without pretrained embedding weights took longer to train than the one with pretrained weights. Over a period 10 epochs, the model utilizing pretrained word embeddings reached its last saved epoch at epoch 5, whereas the model without embeddings reached its last saved epoch at epoch 2. Figure. 13 illustrates the training time per epoch for each model.

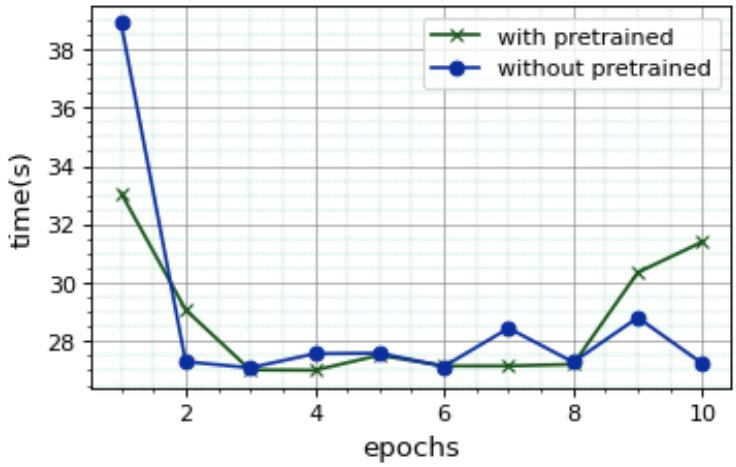


Figure 13. Bi-LSTM models' epoch training time.

Figure 13 shows that the model with pretrained embeddings completed the first epoch in fewer seconds compared to the model without pretrained embedding weights. However, after epoch 8, the training time per epoch was shorter for the model without pretrained embeddings. Figure 14 presents the training and validation loss curves for each model.

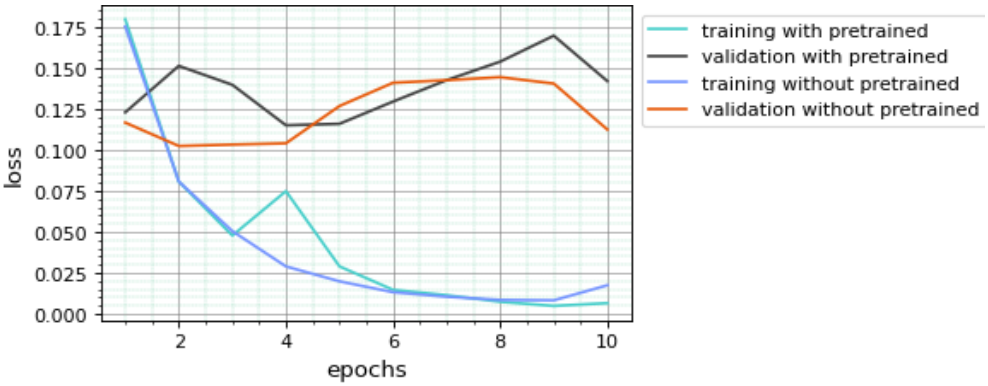


Figure 14. Training and validation losses for models.

Figure 14 shows the training and validation loss curves for two Bi-LSTM models; one using pretrained word embeddings and the other with randomly initialized weights. Initially, both models show a decrease in training and validation losses, indicating improved performance as the models learn from the data. However, the model with pretrained started with low validation loss, suggesting faster leaning of features. Figure 15 shows the training and validation loss of the model using pretrained word embedding with the other one that does not use pre trained word embeddings.

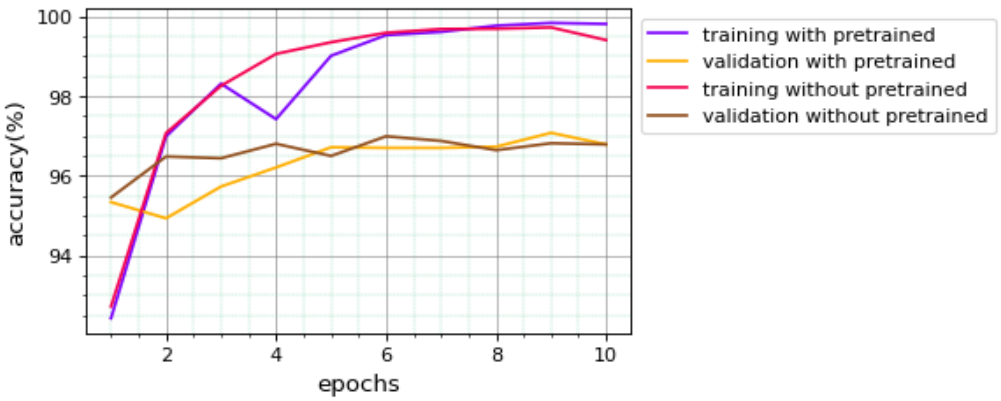


Figure 15. Training and validation accuracies for models.

Figure 15 show the training and validation accuracies across epochs for the same two Bi-LSTM models. The model with pretrained embeddings quickly reaches higher accuracy levels, achieving over 96% validation accuracy by epoch 5. Table 11 shows the best model's accuracy and losses after the model has been evaluated using the testing dataset.

Table 11. Models' losses and accuracies.

Model Name	Loss	Accuracy
With pretrained embedding weights.	0.102	96.51
Without pretrained word embeddings	0.104	96.23

Table 11 shows the test loss and accuracy of the best-performing versions of each Bi-LSTM model. The model with pretrained embeddings achieved a bit better test loss of 0.102 and a test accuracy of 96.51%, while the model without pretrained embeddings had a test loss of 0.104 and accuracy of 96.23%. Figure 16 shows the confusion matrix of a model that was trained using pretrained embedding matrix.

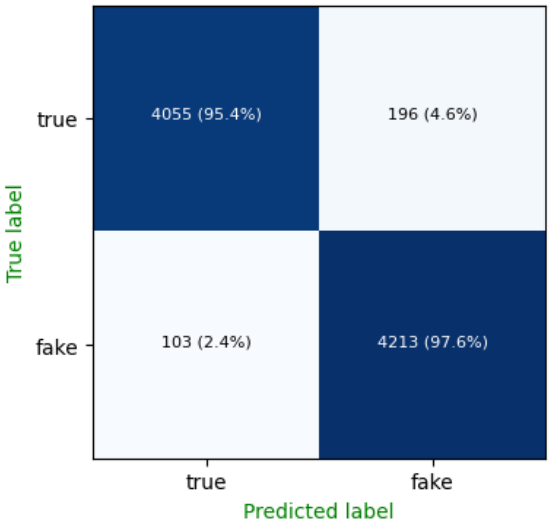


Figure 16. Confusion matrix of the model with pretrained embedding weights.

Figure 16 shows the confusion matrix for the Bi-LSTM model that utilized pretrained domain-specific embeddings. The model correctly classified 4,055 instances of true news (95.4%) and 4,213 instances of fake news (97.6%), showing high accuracy in both classes. The model misclassified 196 true news instances as fake (4.6%) and 103 fake news instances as true (2.4%). Figure 17 shows the confusion matrix of the model that was trained without pretrained domain specific embeddings.

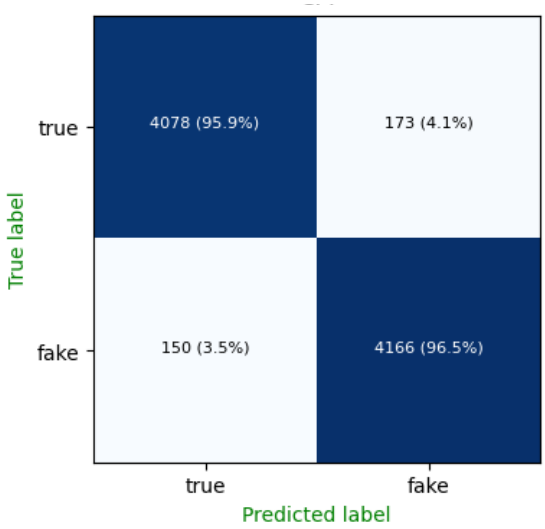


Figure 17. Confusion matrix of the model without pretrained embedding weights.

Figure 17 displays the confusion matrix for the Bi-LSTM model without pretrained embeddings. This model correctly identified 4,078 true news instances (95.9%) and 4,166 fake news instances (96.5%), which is slightly lower in performance compared to the model with pretrained embeddings. It incorrectly labelled 173 true news instances as fake (4.1%) and 150 fake news instances as true (3.5%). Figure 18 presents the classification report for the model with pretrained embedding weights.

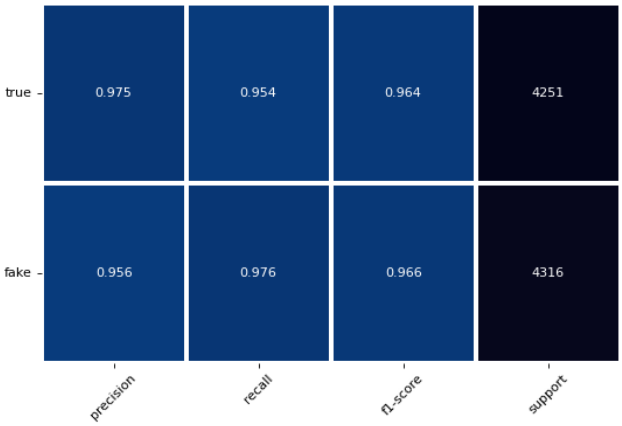


Figure 18. Classification Report of the model with pretrained embedding weights.

This model demonstrates high performance across various metrics, with a precision of 0.975, a recall of 0.954, and an F1-score of 0.964 for detecting true news. For fake news detection, the model achieves a precision of 0.956, recall of 0.976, and an F1-score of 0.966 as shown by Figure 18. Figure 19 displays the classification report for the model without pretrained embedding weights.

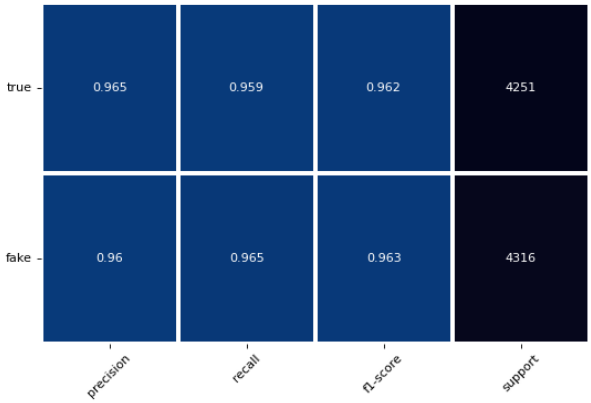


Figure 19. Classification Report of the model without pretrained embedding weights.

From Figure 19, we can see that this model still performs well, achieving a precision of 0.965, recall of 0.959, and F1-score of 0.962 for true news, as well as a precision of 0.960, recall of 0.965, and F1-score of 0.963 for fake news, its metrics are slightly lower compared to the model with pretrained embeddings.

The comparative results of the Bi-RNN and Bi-LSTM models show that domain-specific embeddings provide significant improvements in model performance, especially in fake news detection. Training times vary between models, with the Bi-RNN model requiring a total training time of 13 minutes and 23 seconds over 20 epochs, while the Bi-LSTM models complete training in less than five minutes. This is because our Bi-RNN model had more trainable parameters compared to the ones which was created using Bi-LSTM architecture.

The Bi-LSTM model using pre-trained domain-specific embeddings achieved optimal performance in epoch 5, compared to epoch 2 for the model without embeddings. This faster convergence may be due to the pre-trained embeddings, which allow the model to start with a baseline understanding of news-specific language patterns, as demonstrated by previous studies highlighting the value of contextual embeddings in specialized domains [3], [17]

The slightly higher accuracy of the pre-trained Bi-LSTM model (96.51%) compared to the model without embeddings (96.23%) further highlights the benefits of domain-specific embeddings. These embeddings improve semantic understanding, which is essential for detecting subtle linguistic cues associated with fake news [2]. Additionally, the lower loss and higher F1 score for the pre-trained model indicate improved classification accuracy and recall, confirming the findings of [29] that specialized embeddings improve the sensitivity and specificity of NLP models in misinformation detection tasks.

The training and validation losses across epochs reflect the learning dynamics of each model. The pre-trained Bi-LSTM model shows stable training and validation losses early on as it leverages existing embeddings to quickly identify relevant patterns. In contrast, the model without pre-trained embeddings initially fluctuates

in loss values, reflecting the additional time required to learn the linguistic structures from scratch.

The gradual decrease in loss across successive epochs of this model suggests that, despite their effectiveness, models without domain-specific embeddings take longer to generalize effectively across classes. The Bi-RNN model, which is primarily used to generate embeddings, shows significant variation in training time per epoch, with epochs averaging around 40 seconds but increasing in later epochs. This variability can be attributed to backpropagation adjustments when the model processes dense embedding matrices, a computationally demanding process [34].

Notably, early stopping was used at epoch 5, allowing the model to avoid overfitting. The consistent performance observed with this approach suggests that integration-specific training facilitates early convergence, consistent with recent findings by [8] on efficient resource utilization in NLP integration tasks.

The confusion matrices for the two Bi-LSTM models show that the model with pre-trained embeddings achieved lower classification errors across classes. It successfully classified 97.6% of fake news instances and 95.4% of true news, highlighting high accuracy in identifying both categories. In contrast, the model without pre-trained embeddings showed slightly higher error rates, with 96.5% accuracy in classifying fake news and 95.9% in classifying real news.

This finding is consistent with the work of [10], who noted that domain-specific embeddings improve detection accuracy by capturing task-relevant semantics. In addition to the accuracy metrics, the classification reports further support the superiority of the pre-trained model. The higher precision, recall, and F1 scores across all categories of the pre-trained model highlight the importance of domain-specific embeddings for distinguishing nuanced linguistic signals. For example, the F1 score of 0.966 for the fake news class in the pre-trained model illustrates how embeddings trained on news data improve the model's ability to capture subtle and obvious indicators of misinformation [44].

Integrating domain-specific embeddings significantly reduces training time while improving model performance. By providing pre-trained representations, the model can avoid the need to independently learn contextual relationships between words, thereby facilitating faster convergence and improved accuracy [15]. This finding supports previous work by [20], who demonstrated that domain-specific integration leads to more efficient learning and lower error rates.

Despite their benefits, domain-specific integration still poses challenges in terms of computational demands. Initial training of the Bi-RNN model requires significant processing resources, as evidenced by variable epoch times. Additionally, managing the complexity of the integration process can increase the risk of overfitting, especially in specialized tasks with small datasets [26]. Regularization techniques, such as dropout layers with a 50% dropout rate as implemented here, are essential to maintain the generalizability of the model [8].

The results of this study confirm that domain-specific integration plays an important role in improving fake news detection. The Bi-LSTM model with pre-trained embeddings not only achieves higher classification accuracy but also shows better model stability, reducing both training time and computational requirements. These results highlight the effectiveness of custom integration in specialized NLP

applications, as suggested by studies in various domains such as radiology and financial analysis [2], [38].

D. Conclusion

This study presents a comprehensive analysis of domain-specific word embeddings to improve the accuracy of fake news detection, focusing on the comparative performance of models with and without pre-trained news embeddings. The results confirm that domain-specific embeddings improve the model's ability to classify fake and real news by enabling more nuanced semantic understanding, especially in specialized textual contexts.

The Bi-LSTM model with pre-trained embeddings shows higher accuracy (96.51%) and higher training efficiency than the model without pre-trained embeddings, highlighting the advantage of custom embeddings in reducing computational requirements and classification errors. The improved performance of domain-specific embeddings confirms their importance for specialized NLP tasks where subtle linguistic differences are essential. The embeddings are tailored to the news domain allowing the model to capture context-sensitive word relationships and reducing the need for extensive training, corroborating the findings of researchers such as [2] and [3].

However, the study also highlights the computational intensity involved in training these embeddings, suggesting that resource optimization techniques are necessary for practical implementation in real-time systems. Overall, this study contributes to the advancement of fake news detection by demonstrating that domain-specific embeddings not only improve accuracy but also provide more stable training dynamics.

Future research could focus on optimizing embedding generation processes and explore the integration of multimodal features, such as images and social metadata, to further improve model performance in fake news detection applications.

E. Acknowledgment

We would like to thank the creators of the datasets used in this study, especially [23], who provided the “News Article Classification Dataset for Natural Language Processing and Machine Learning,” and [24], who created the “Fake News Detector Tool.” dataset, both of which are available on Kaggle. These datasets were invaluable in training and evaluating our models, and their availability greatly contributed to the progress and results of this study.

F. References

[1] M. Al-alshaqi, D. B. Rawat, and C. Liu, “Ensemble Techniques for Robust Fake News Detection: Integrating Transformers, Natural Language Processing, and Machine Learning,” *Sensors*, vol. 24, no. 18, Sep. 2024, doi: 10.3390/s24186062.

[2] T. L. Chen *et al.*, “Domain specific word embeddings for natural language processing in radiology,” *J Biomed Inform*, vol. 113, Jan. 2021, doi: 10.1016/j.jbi.2020.103665.

[3] C. O. Truică and E. S. Apostol, "It's All in the Embedding! Fake News Detection Using Document Embeddings," *Mathematics*, vol. 11, no. 3, Feb. 2023, doi: 10.3390/math11030508.

[4] Y. Chopra, P. Kaushik, S. P. S. Rathore, and P. Kaur, "Uncovering Semantic Inconsistencies and Deceptive Language in False News Using Deep Learning and NLP Techniques for Effective Management," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 8s, pp. 681–692, Aug. 2023, doi: 10.17762/ijritcc.v11i8s.7256.

[5] P. Bazmi, M. Asadpour, A. Shakery, and A. Maazallahi, "Entity-centric multi-domain transformer for improving generalization in fake news detection," *Inf Process Manag*, vol. 61, no. 5, Sep. 2024, doi: 10.1016/j.ipm.2024.103807.

[6] A. H. Hoti, M. H. Hoti, H. Hoti, and A. Salihu, "Identifying Fake News written on Albanian language in social media using Naïve Bayes, SVM, Logistic Regression, Decision Tree and Random Forest algorithms," in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, Jun. 2022, pp. 1–6. doi: 10.1109/MECO55406.2022.9797147.

[7] H. Noor, J. Ahmad, A. Haider, F. Nasim, and A. Jaffar, "A Machine Learning Sentiment Analysis Approach on News Headlines to Evaluate the Performance of the Pakistani Government", doi: 10.56979/702/2024.

[8] T. Perumal, N. Mustapha, R. Mohamed, and F. M. Shiri, "A Comprehensive Overview and Comparative Analysis on Deep Learning Models," *Journal on Artificial Intelligence*, vol. 6, no. 1, pp. 301–360, 2024, doi: 10.32604/jai.2024.054314.

[9] A. Hassan and A. Mahmood, "Convolutional Recurrent Deep Learning Model for Sentence Classification," *IEEE Access*, vol. 6, pp. 13949–13957, Mar. 2018, doi: 10.1109/ACCESS.2018.2814818.

[10] S. Raza and C. Ding, "Fake news detection based on news content and social contexts: a transformer-based approach," *Int J Data Sci Anal*, vol. 13, no. 4, pp. 335–362, May 2022, doi: 10.1007/s41060-021-00302-z.

[11] P. M. Subhash, D. Gupta, S. Palaniswamy, and M. Venugopalan, "Fake News Detection Using Deep Learning and Transformer-Based Model," in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jul. 2023, pp. 1–6. doi: 10.1109/ICCCNT56998.2023.10308352.

[12] Y. Wang *et al.*, "EANN: Event adversarial neural networks for multi-modal fake news detection," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, Jul. 2018, pp. 849–857. doi: 10.1145/3219819.3219903.

[13] X. Zhou and R. Zafarani, "A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities," *ACM Comput Surv*, vol. 53, no. 5, Sep. 2020, doi: 10.1145/3395046.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality."

[15] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Stroudsburg, PA, USA: Association

for Computational Linguistics, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

[16] T. Cohen Biomedical, H. Informatics, and D. Widdows, "Bringing Order to Neural Word Embeddings with Embeddings Augmented by Random Permutations (EARP)," 2018.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.04805

[18] S. A. Aljawarneh and S. A. Swedat, "Fake News Detection Using Enhanced BERT," *IEEE Trans Comput Soc Syst*, vol. 11, no. 4, pp. 4843–4850, Aug. 2024, doi: 10.1109/TCSS.2022.3223786.

[19] J. Lee *et al.*, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020, doi: 10.1093/bioinformatics/btz682.

[20] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text," Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.10676

[21] Y. Yang, M. C. S. UY, and A. Huang, "FinBERT: A Pretrained Language Model for Financial Communications," Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.08097

[22] W. Y. Ayele, "Adapting CRISP-DM for Idea Mining A Data Mining Process for Generating Ideas using a Textual Dataset," 2020. [Online]. Available: www.ijacsa.thesai.org

[23] V. Banuprakash, "News Articles Classification Dataset for NLP & ML." Accessed: Nov. 09, 2024. [Online]. Available: https://www.kaggle.com/datasets/banuprakashv/news-articles-classification-dataset-for-nlp-and-ml/

[24] B. Jikadara, "Fake News Detection." Accessed: Nov. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/bhavikjikadara/fake-news-detection

[25] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, Oct. 2018, doi: 10.1016/j.neunet.2018.07.011.

[26] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0192-5.

[27] H. Tang, S. Kamei, and Y. Morimoto, "Data Augmentation Methods for Enhancing Robustness in Text Classification Tasks," *Algorithms*, vol. 16, no. 1, Jan. 2023, doi: 10.3390/a16010059.

[28] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[29] R. K. Kaliyar, A. Goswami, and P. Narang, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *Multimed Tools Appl*, vol. 80, no. 8, pp. 11765–11788, Mar. 2021, doi: 10.1007/s11042-020-10183-2.

[30] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake news using text classification," *SECURITY AND PRIVACY*, vol. 1, no. 1, Jan. 2018, doi: 10.1002/spy2.9.
[31] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength natural language processing in python." Zenodo, Honolulu, HI, USA, 2020.
[32] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.
[33] J. Brownlee, *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. 2020.
[34] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980
[35] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html.
[36] Francois. Chollet, *Deep Learning with Python by Francois Chollet*. Manning Publications : [distributed by] Skillsoft Books, 2019.
[37] D. M. W. Powers and Ailab, "EVALUATION: FROM PRECISION, RECALL AND F-MEASURE TO ROC, INFORMEDNESS, MARKEDNESS & CORRELATION."
[38] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding." [Online]. Available: https://github.com/zihangdai/xlnet
[39] A. Vaswani *et al.*, "Attention Is All You Need."
[40] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Commun ACM*, vol. 64, no. 3, pp. 107–115, Mar. 2021, doi: 10.1145/3446776.
[41] J. Howard and S. Ruder, "Universal Language Model Fine-tuning for Text Classification," Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.06146
[42] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in Pre-Training Distributed Word Representations," Dec. 2017, [Online]. Available: http://arxiv.org/abs/1712.09405
[43] Z. Khanam, B. N. Alwasel, H. Sirafi, and M. Rashid, "Fake News Detection Using Machine Learning Approaches," *IOP Conf Ser Mater Sci Eng*, vol. 1099, no. 1, p. 012040, Mar. 2021, doi: 10.1088/1757-899x/1099/1/012040.
[44] D. Wu, Z. Tan, H. Zhao, T. Jiang, and N. Geng, "Domain- and category-style clustering for general fake news detection via contrastive learning," *Inf Process Manag*, vol. 61, no. 4, Jul. 2024, doi: 10.1016/j.ipm.2024.103725.