International Journal of Scientific Research and Management (IJSRM)
||Volume||10||Issue||06||Pages||EC-2022-929-941||2022||
Website: www.ijsrm.net ISSN (e): 2321-3418
DOI: 10.18535/ijsrm/v10i6.ec06

Robust Detection of Fake News Using LSTM and GloVe Embeddings

[1]Alex Lee, [2]Xiang Chen, [3]Ivy Wood

1. Department of Information System Stevens Institute of Technology
2. Department of Computer Science Boston University
3. Department of data science MathWorks

Abstract
The pervasive issue of fake news on digital platforms poses a significant threat to public opinion and trust in me- dia. This paper addresses the problem of fake news detection by leveraging advanced natural language processing (NLP) techniques and deep learning models. Utilizing the ISOT Fake News Dataset, which comprises balanced samples of verified and fake news articles, we develop and evaluate two primary models: a Long Short-Term Memory (LSTM) network and a Convolutional Neural Network (CNN). The LSTM model employs pre-trained GloVe embeddings, fol- lowed by LSTM layers and a fully connected layer for clas- sification, while the CNN model incorporates convolutional layers, max-pooling, and dropout for comparative analy- sis. Extensive pre-processing and exploratory data analysis (EDA) were conducted to clean the data and understand its characteristics. Our results demonstrate that the LSTM model outperforms the CNN model, achieving an accuracy of 99.58% on the test set. However, the high performance raises concerns about dataset biases, suggesting the need for more diverse and challenging datasets to ensure model ro- bustness. Future work will focus on adversarial training and explainability techniques to enhance the model's resilience and interpretability.

1. Problem Framing

The proliferation of fake news on digital platforms has become a significant issue, impacting public opinion and trust in media. Accurately detecting and classifying fake news is essential to mitigate its adverse effects (1). This research addresses the problem of fake news detection by leveraging machine learning models to distinguish between fake and verified news articles. Using the ISOT Fake News Dataset, which includes balanced samples of both true and fake news, this study aims to develop and evaluate robust models. The focus is on employing advanced natural lan- guage processing techniques and deep learning architectures to achieve high accuracy and reliability in fake news clas- sification.

2. Introduction

The rise of fake news poses a critical threat to the integrity of information disseminated through digital plat- forms, influencing public perception and decision-making processes. Consequently, detecting and preventing fake news have garnered significant attention in both academic research and practical applications. This research aims to develop and evaluate robust machine learning models that distinguish between fake and verified news articles, utilizing advanced natural language processing (NLP) techniques and deep learning architectures. The study implemented the ISOT Fake News Dataset, which comprises a balanced collection of authentic and fake news articles. The dataset includes article titles, content, subjects, and publication dates. Exten- sive pre-processing steps were undertaken to clean the data, including removing HTML tags, URLs, hashtags, punctua- tion, and irrelevant characters, unifying date formats, and eliminating duplicates. Exploratory Data Analysis (EDA) was conducted to gain insights into the dataset's distribution and characteristics, employing visualizations such as word clouds and n-grams to highlight distinguishing features be- tween genuine and fake news articles.

Two primary models were implemented and evaluated in this research: a Long Short-Term Memory (LSTM) net- work and a Convolutional Neural Network (CNN). The LSTM model, known for its ability to

capture sequential dependencies in text, was selected for its effectiveness in processing natural language data. The CNN model, recog- nized for its proficiency in identifying local patterns, was a comparative approach to evaluate its performance in text classification tasks. The LSTM model utilized pre-trained GloVe embeddings to harness contextual word representa- tions, followed by LSTM layers and a fully connected layer for classification. In contrast, the CNN model incorporated a one-dimensional convolutional layer, max-pooling, dropout, and a fully connected layer. Both models were trained and evaluated on the dataset, with performance metrics such as accuracy, precision, recall, and F1-score used to assess their effectiveness.

Results indicated that the LSTM model outperformed the CNN model, achieving a high accuracy of 99.58% on the test set. However, the exceptional performance raised concerns regarding the dataset's robustness and potential bi- ases. Further analysis suggested more diverse and challeng- ing datasets, including adversarial examples, to ensure the
model's resilience and applicability in real-world scenarios. This research demonstrates the potential of deep learning models in detecting fake news while highlighting the im- portance of dataset quality and diversity. Future work will enhance model robustness through adversarial training and explore explainability techniques such as Lime and SHAP values to understand better and interpret model predictions.

2.1. Dataset Description

The dataset used in this research is the ISOT Fake News Dataset, curated by the Information Security and Object Technology (ISOT) research group at the Univer- sity of Victoria. This dataset comprises two distinct news

Table 1. Summary of News Articles

Title	Text	Subject	Date
As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

articles: accurate news and fake news. The articles are sourced from reputable news websites such as Reuters, en- suring their credibility and authenticity. Conversely, the fake news articles are collected from various unreliable websites flagged by organizations such as Politifact and Wikipedia for spreading misinformation. Each article in the dataset includes attributes such as the title, textual content, subject, and publication date. The dataset is balanced, containing an equal number of genuine and fake news articles, which is crucial for training machine learning models without introducing class bias. This comprehensive dataset provides a robust foundation for developing and evaluating fake news detection models, allowing for thorough analysis and accurate classification of news articles.

The table t1 presents a concise overview of news articles, highlighting four main attributes for each entry: the title, text, subject, and publication date. It contains examples of articles from a political news dataset, showing brief excerpts of article titles and texts alongside their corresponding sub- jects and publication dates. The table

effectively summarizes
key information, allowing for quick reference and compar-ison of articles.

3. Data Preprocessing

Data Preprocessing

The ISOT Fake News Dataset's preprocessing was a critical step to ensure the quality and consistency of the data used for training and evaluating the machine learning models. The dataset comprised two subsets: verified news articles sourced from reputable websites like Reuters and fake news articles collected from unreliable sources flagged by fact-checking organizations. Each article contained at- tributes such as title, text, subject, and publication date. Initially, genuine and fake news datasets were loaded into pandas DataFrames, followed by a check for missing values, duplicates, and the format of various fields. It was noted that accurate news articles often began with a specific pattern, "CITY (Reuters) -," which was removed to prevent bias in the model. Comprehensive cleaning processes were applied to the text data, including removing HTML tags, URLs, and hashtags, stripping special characters and punctuation, con- verting all text to lowercase, and removing common Englishstopwords and specific irrelevant words and characters usingthe NLTK library.

Table 2. Word Occurrences In The Dataset

Word	Occurrences
said	72025
U.S.	38271
Trump	36461
would	31330
-	31059
said.	21582
(Reuters)	21239
President	17112
also	15703
United	15030

The dates in the dataset were standardized to a consistent datetime format, and incorrectly formatted dates or those containing irrelevant information (e.g., URLs) were dis- carded. The title and text of each article were concatenated into a single column, creating a unified textual represen- tation for each article, which served as the primary input for the NLP models. A binary label was assigned to each article, with '1' indicating verified news and '0' indicating fake news. The cleaned and processed authentic and fake news datasets were then concatenated into a single dataset, which was shuffled and split into training, validation, and testing subsets. The text data was tokenized into words using NLTK's word tokenizer, and these tokens were converted into sequences of indices corresponding to a vocabulary built from the training data. The sequences were padded to a maximum length to ensure uniform input size for the models. A vocabulary was constructed from the training dataset, mapping each unique word to an index, which was essential for embedding the text data into numerical vectors using pre-trained GloVe embeddings.

Through these preprocessing steps, the dataset was trans- formed into a structured and clean format suitable for feeding into the machine learning models. This rigorous preprocessing ensured that the models received high-quality input data, enhancing their performance and reliability in detecting fake news. The table 2 in the Dataset" presents the frequency of specific words within the dataset. It lists the words alongside their respective occurrences, highlighting the most frequently appearing terms. For instance, the word "said" appears 72,025 times, while "U.S." and "Trump" occur 38,271 and 36,461 times, respectively. This table provides a clear overview of the most common words foundin the dataset. The figure 1 showing the distribution of

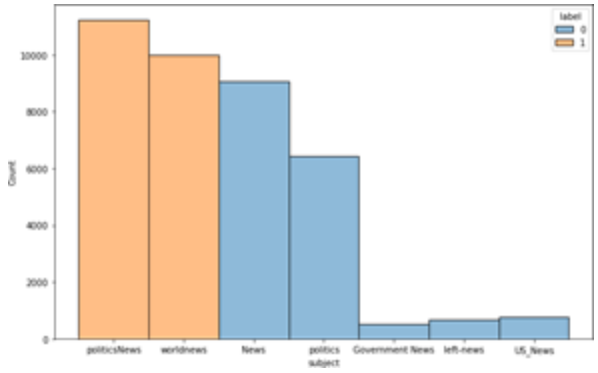

Figure 1. News Subjects

news articles across various subjects, categorized by labels indicating fake (0) and verified (1) news. The subjects on the x-axis include "politicsNews," "worldnews," "News," "pol- itics," "Government News," "left-news," and "US News," while the y-axis represents the count of articles. The chart reveals that "politicsNews" and "worldnews" have the high- est counts of articles, with both categories predominantly consisting of verified news (1). The "News" category, on the other hand, shows a substantial number of articles, mainly labeled as fake news (0). Other subjects such as "pol- itics," "Government News," "left-news," and "US News" have fewer articles, with "politics" and "Government News" exhibiting a higher count of fake news (0). Overall, the chart provides a clear visual comparison of the prevalence of fake and verified news across different subjects. The figure 2 compares the average character length of true and false news articles. The y-axis represents the average character length, while the x-axis shows the labels "True" and "False." The chart indicates that both true and false news articles have similar average character lengths, with no significant differ- ence between the two. This suggests that character length alone may not be a distinguishing factor in identifying fake news. The pair of histograms 3 compares the distribution of article lengths, measured by the number of characters, for true and false news articles. Both true and false news articles generally have fewer than 5000 characters, indicating no striking difference in overall length. However, the histogram for false news shows a distinct peak between 2000 and3000 characters, suggesting that fake news articles more

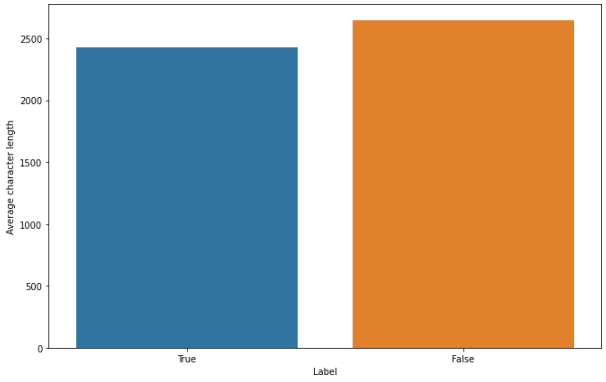

Figure 2. Class Label

frequently fall within this range compared to true news. In contrast, the length distribution for true news articles is more evenly spread out with multiple smaller peaks. This highlights a subtle difference in the typical lengths of true versus false news articles. The pair of histograms 4compares

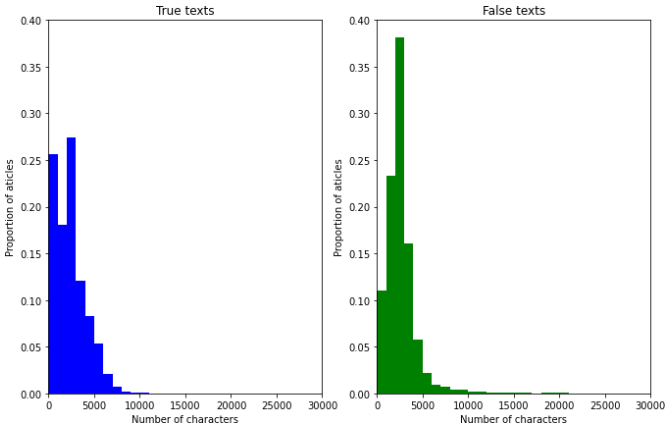

Figure 3. Number of Characters

the distribution of article lengths, measured by the number of words, for true and false news articles. Both true and false news articles generally contain fewer than 1500 words, with a similar distribution pattern. The majority of articles, both true and false, have fewer than 1000 words. However, the histogram for false news shows a noticeable peak around 500-600 words, suggesting that fake news articles more frequently fall within this range compared to true news. In contrast, the length distribution for true news articles is slightly more spread out with smaller peaks. This highlights a subtle difference in the typical lengths of true versus false news articles in terms of word count.

Following this analysis, the articles underwent a cleaning process to remove HTML tags, URLs, hashtags, Twitter han- dles, and square brackets, particularly prevalent in the fake news dataset. Additionally, stopwords and punctuation were removed to standardize the text and enhance the accuracy of the subsequent analysis. The word cloud 5 visualization rep-

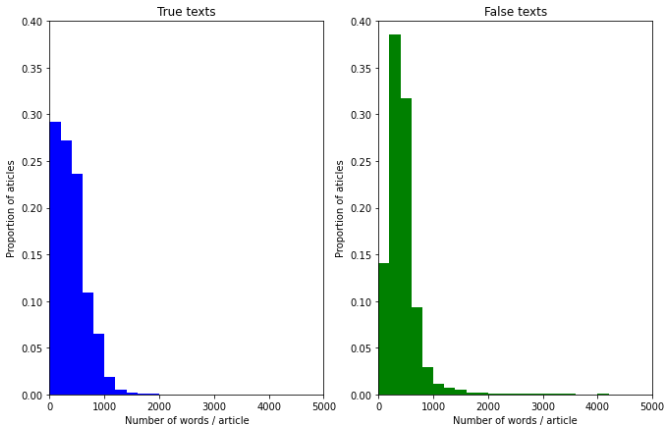

Figure 4. Number of words/Articles

resents the most frequently occurring words in the true news articles. Prominent words like “United,” “States,” “Trump,” “White,” “House,” and “North Korea” appear larger and more centrally positioned, indicating their high frequency within the dataset. Other notable words include “President,” “government,” “prime minister,” “said,” “statement,” and “support.” This visualization helps to quickly identify key topics and entities commonly discussed in true news articles, with a significant focus on political figures and international relations. The presence of words like “Obama,” “Clinton,” and “Reuters” further emphasizes the political and news-centric nature of the true news dataset. The word cloud 6

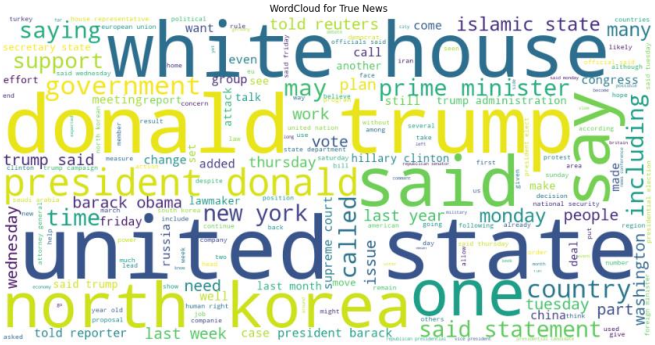

Figure 5. Word Cloud of True News

visualization for false news articles highlights the most fre- quently occurring words within the dataset. Prominent words such as “Donald,” “Trump,” “said,” “one,” “time,” “fea- tured,” “image,” “even,” and “say” appear larger, indicating their high frequency. Other notable words include “Repub- lican,” “people,” “Clinton,” “Obama,” “house,” “American,” “country,” and “via.” The word cloud emphasizes a strong focus on political figures and topics, similar to the true news word cloud, but also includes a higher occurrence of sensa- tional and visually oriented terms like “featured,” “image,” and “via.” This suggests that false news articles may often include more visual content and sensational language. The presence of words like “Twitter,” “Getty,” and “pic” further underscores the focus on media and visual elements in the false news dataset.

Figure 6. Word Cloud of False News

3.1. Frequency Analysis of Unigrams in the TextCorpus

In figure 7, we utilized the CountVectorizer from the sklearn.feature_extraction.text li- brary to compute the most frequent unigrams (single words) within a corpus of text. By setting the ngram_range to (1, 1), we focused on extracting unigrams. The function get_ngrams was employed to transform the corpus into a matrix of token counts and then sum these counts to obtain the total occurrences of each unigram. The resulting list of unigrams and their frequencies was sorted in descending order to identify the top *n* most frequent unigrams. To visualize these findings, the plot_ngrams function was used to create a line plot. This plot displays the occurrence count on the x-axis and the most frequent unigrams on the y-axis, highlighting the words with the highest frequenciesin the corpus.

The resulting plot shows that the word “said” has the highest occurrence, followed by “Trump,” “would,” “presi- dent,” and “state,” among others. This visualization helps in understanding the prominent terms used in the text, which can be crucial for text analysis and further natural lan- guage processing tasks. In 2 Gram 8 analysis, we extended the previous unigram frequency analysis to bigrams (two- word combinations) using the same CountVectorizer from the sklearn.feature_extraction.text li- brary. By setting the ngram_range to (2, 2), we fo- cused on extracting bigrams from the text corpus. The get_ngrams function was employed to transform the corpus into a matrix of bigram counts and then sum these counts to obtain the total occurrences of each bigram. The resulting list of bigrams and their frequencies was sorted in descending order to identify the top

n most frequent bigrams.

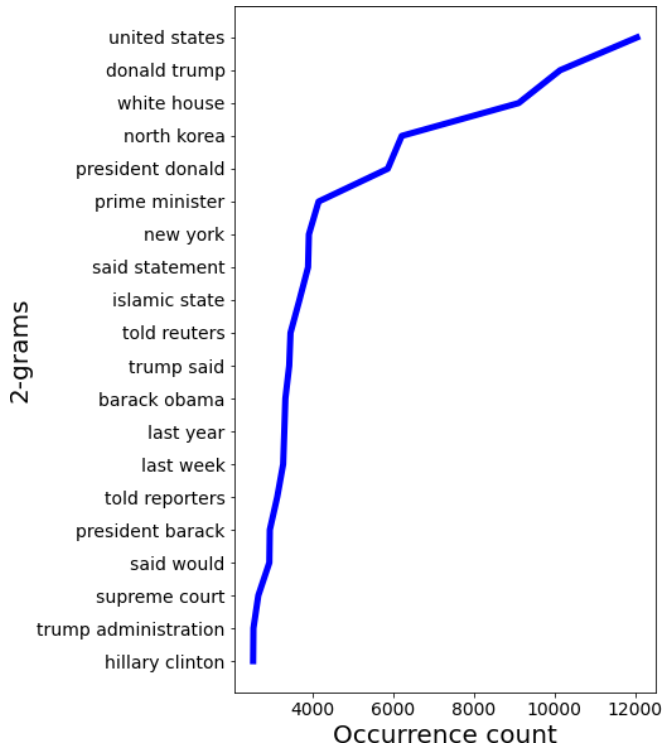

Figure 7. Frequency Analysis of Unigrams(1 Gram)

The visualization created using the plot_ngrams function displays the occurrence count on the x-axis and the most frequent bigrams on the y-axis. This plot highlights the prominent two-word combinations used in the corpus.

The resulting plot shows that the bigram “united states” has the highest occurrence, followed by “donald trump,” “white house,” “north korea,” and “president donald,” among others. This visualization helps in understanding the key two-word phrases that are commonly used in the text, pro- viding deeper insights into the patterns and relationshipswithin the corpus.

4. Model Prediction

For the model prediction phase, we utilized PyTorch and several essential libraries. Initially, we defined a function to split the dataset into training, validation, and test sets based on specified proportions. This splitting ensured that the data was shuffled and evenly distributed across the sets. We then created a function to return the distribution of true and fake news in the dataset, which helped in maintaining a balanced dataset. We designed a custom dataset class inheriting from PyTorch’s Dataset module. This class facilitated building the vocabulary from the training data, mapping words to indices, and vice versa. Each article was tokenized using

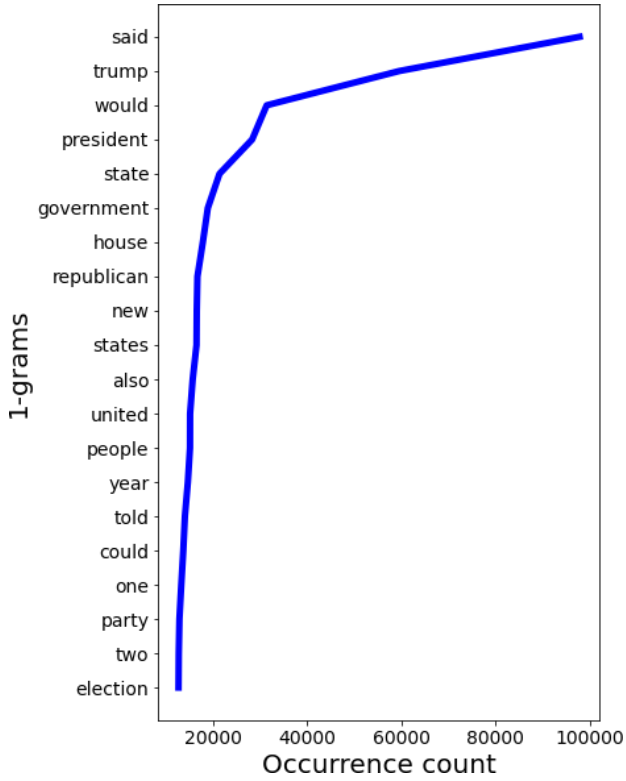

Figure 8. Frequency Analysis of Unigrams(2 Gram)

NLTK, converted to indices, and then transformed into tensors. To ensure uniformity, sequences were padded to a predetermined length. This preprocessing step was crucial in preparing the data for the subsequent training and evaluationof the model.

4.1. Embeddings & Model Design

n the Embeddings & Model Design phase, we utilized the pre-trained GloVe embeddings from the ”glove-wiki-gigaword-300” model provided by the Gensim library. The GloVe model consists of word vectors with 300 dimensions, trained on a large corpus from Wikipedia. To adapt these embeddings to our vocabulary, we implemented a function get_glove_adapted_embeddings.

This function first maps our vocabulary words to their corresponding GloVe embeddings. It creates an embedding matrix where each row corresponds to a word in our vocabu- lary, and each column represents one of the 300 dimensions of the GloVe embeddings. The resulting embedding matrix, GloveEmbeddings, ensures that our model leverages the rich semantic information encoded in the GloVe vectors, enhancing its ability to understand the context and meaning of words within the articles. Mathematically, for each word w in our vocabulary with index i, the embedding matrix entry $\mathbf{E}[\boldsymbol{i}]$ is set to the GloVe vector of w. $\mathbf{E}[\boldsymbol{i}] = \text{GloVe}(w)\ \forall w \in \text{vocabulary}$ (1)

5. LSTM Model Definition

Our Long Short-Term Memory (LSTM) model is de- signed to handle the sequential nature of text data, leveraging the GloVe embeddings for enhanced semantic un- derstanding. The model is defined within the LSTMModel class, which inherits from PyTorch’s nn.Module.

The__init__method initializes the model with sev- eral key components:

- **Embedding Layer:** If pre-trained GloVe embed- dings are provided, they are loaded into an em- bedding layer, which can be optionally fine-tuned. Otherwise, a randomly initialized embedding layer is created. This layer transforms input words into dense vectors of fixed size (embedding_dim).
- **LSTM Layer:** A two-layer LSTM is created with

a specified hidden dimension (hidden_dim). This recurrent layer processes the sequence of embed- dings and captures temporal dependencies.

- **Linear Layer:** A fully connected linear layer maps the concatenated hidden states of the LSTM to the output space. The hidden states from the last two LSTM layers are concatenated, resulting in a feature vector of size $2 \times$ hidden_dim.

The forward method defines the forward pass through the model:

1) The input sequence is passed through the embed- ding layer to obtain the word vectors.
2) The embeddings are processed by the LSTM layers,which output the hidden states and cell states.
3) The hidden states from the last two LSTM layers are concatenated and passed through the linear layer.
4) The output of the linear layer is squeezed to removeany extra dimensions, producing the final predic- tion.

Mathematically, the forward pass can be expressed as:

$$\mathbf{H} = \text{LSTM}(\mathbf{E}) \text{ and } \mathbf{y} = \text{Linear}([\mathbf{H}_{\text{last}}]) \quad (2)$$

where $\mathbf{E}$ is the matrix of embeddings, $\mathbf{H}_{\text{last}}$ are the last hidden states from the LSTM layers, and $\mathbf{y}$ is the output prediction.

This architecture enables the model to capture both the contextual and sequential information inherent in the news articles, making it well-suited for the fake news detection task.

5.1. Model Training and Evaluation optimizer, loss criterion, and dataloader. During each itera-tion over the dataloader, it performs the following steps:

1) Sets the model to training mode.
2) Resets the gradients of the optimizer.
3) Performs a forward pass to compute predictions.
4) Computes the loss using the defined criterion.
5) Performs a backward pass to compute gradients.
6) Updates the model weights using the optimizer.
7) Calculates the accuracy for the current batch.

This function returns the losses and accuracies for eachbatch within the epoch.

The eval_model function is used for evaluating the model on the validation or test dataset. It sets the model to evaluation mode and performs a forward pass to compute predictions without updating the weights. The function cal- culates the loss and accuracy over all batches and returns these metrics along with the predictions.

The experiment function orchestrates the train- ing and evaluation process over multiple epochs. It it- erates through the specified number of epochs, calling train_epoch for each epoch to train the model and eval_model at the end of each epoch to evaluate per- formance on the validation set. After training, it performs a final evaluation on the test set. This function tracks and prints the training and validation losses and accuracies for each epoch and returns the final metrics and predictions.

By using these functions, we ensure a structured and systematic approach to training and evaluating our LSTM model, providing insights into the model's performance across different stages of the training process.

5.2. Model Training Configuration

For our model training, we chose the Adam optimizer, which is well-suited for handling sparse gradients on noisy problems. The optimizer's parameters, including the learn- ing rate and betas, are defined in the code. Given that our task is binary classification, we opted for the binary cross- entropy loss with logits (nn.BCEWithLogitsLoss()), which integrates a sigmoid activation function internally, making it efficient for binary classification tasks.

The embedding dimension (EMBEDDING_DIM) is set to 300, aligning with the GloVe embeddings used earlier. The vocabulary size (VOCAB_SIZE) is derived from the length of our training word index, and the hidden dimension (HIDDEN_DIM) for the LSTM layers is set to 256. We set the learning rate (learning_rate) at 0.0025 to ensure gradual updates, balancing the convergence speed and sta- bility.

To train and evaluate our LSTM model, we implemented several functions to handle the training process, valida- tion, and final testing. The primary functions involved are train_epoch, eval_model, and experiment.

The train_epoch function manages the training of the model for a single epoch. It takes as input the model,

Table 3. Model Configuration Parameters

Parameter	Value
EMBEDDING_DIM	300
VOCAB_SIZE	len(training word2idx)
HIDDEN_DIM	256
learning_rate	0.0025
num_epochs	10

We then initialize our LSTM model with the specified parameters and enable the use of pre-trained embeddings without fine-tuning. The Adam optimizer is configured with the model's parameters, and the binary cross-entropy loss function is instantiated.

The training process spans 10 epochs (num_epochs), during which we call the experiment function. This function handles the training and validation of the model over the epochs and evaluates it on the test set. The out- puts include training and validation losses and accuracies for each epoch, along with the final test loss, accuracy, and predictions. These metrics are crucial for assessing the model's performance and guiding further refinements.

5.3. Training & Validation Accuracy/Loss

During the training process, the model demonstrated sig- nificant improvement over ten epochs. Initially, in Epoch 1, the training loss began at 0.692 and decreased progressively, resulting in a validation loss of 0.267 and an accuracy of 91.41%. By Epoch 2, the validation loss further decreased to
0.127 with an accuracy of 95.35%. In Epoch 3, the model showed substantial improvement with a validation loss of
0.070 and an accuracy of 97.48%. This trend continued in Epoch 4, achieving a validation loss of 0.040 and an accuracy of 98.71%.

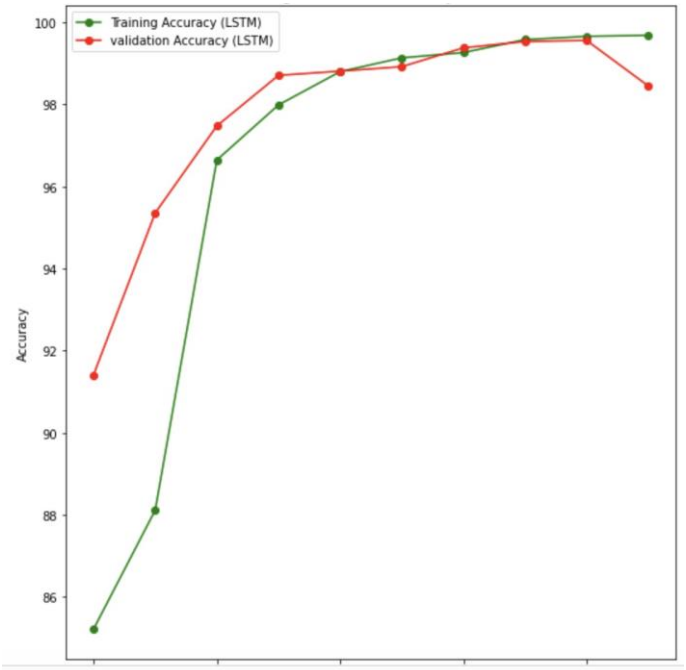

Figure 9. Training & Validation Accuracy (LSTM)

In Epoch 5, the model maintained its performance with a validation loss of 0.033 and an accuracy of 98.81%.

Despite a slight increase in validation loss to 0.038 in Epoch 6, the accuracy remained high at 98.92%. Epoch 7 saw further refinement with a validation loss of 0.019 and an accuracy of 99.38%. The model peaked in Epoch 8 with its lowest validation loss of 0.014 and highest accuracy of 99.54

In Epoch 9, the validation loss slightly increased to 0.013 with an accuracy of 99.56%. By Epoch 10, the validation loss was 0.060, and the accuracy remained robust at 98.45%. Overall, the model's final test results showed a loss of 0.050 and an accuracy of 98.64%, confirming its effectiveness in accurately classifying fake news. Figure 9 and 10 dis-

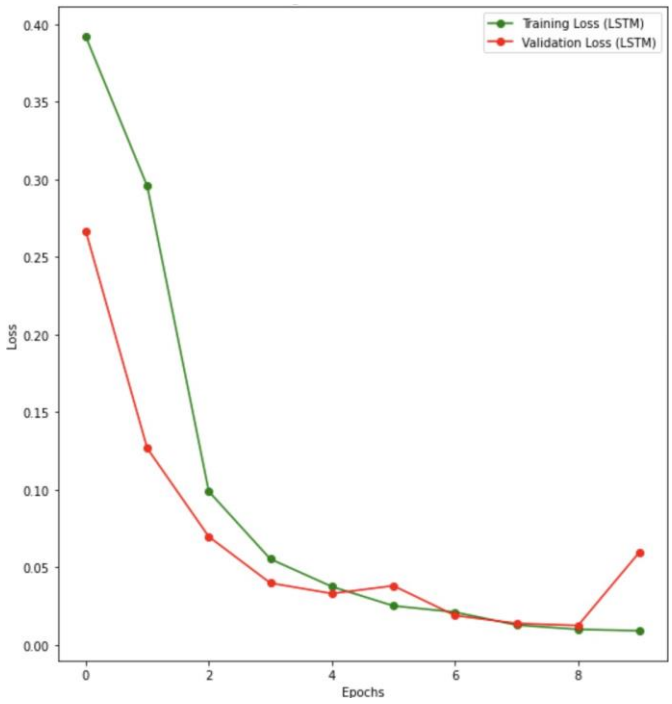

Figure 10. Training & Validation Loss (LSTM)

play the training and validation accuracy and loss curves over 10 epochs for the LSTM model. In the first graph, the accuracy plot, we observe a steady increase in both training and validation accuracy across epochs. Initially, the validation accuracy surpasses the training accuracy, which indicates that the model is learning efficiently from the data. Both accuracies eventually converge, reaching above 98%, suggesting that the model has generalized well withoutoverfitting.

The second graph presents the training and validation loss over the same epochs. Here, both losses show a signif- icant decrease as the training progresses. The training loss decreases consistently, while the validation loss follows a similar pattern but with a slight increase towards the final epochs. This minor increase in validation loss can indicate a slight overfitting, but overall, the losses remain low, con- firming that the model has learned the patterns in the data effectively. These graphs together illustrate the robustness and efficiency of the LSTM model in distinguishing betweentrue and fake news articles.

6. Model Evaluation

In the evaluation phase of our LSTM model, we con- ducted performance analysis using predicted values, a con- fusion matrix, and detailed classification metrics. The pre- dicted values were obtained from the model and processed using a sigmoid function to get probabilities, which were then converted to binary predictions with a threshold of
0.5. The confusion matrix revealed 2142 true positives (fake news correctly predicted as fake), 1671 true negatives (true news correctly predicted as true), 2 false positives (true news incorrectly predicted as fake), and 50 false negatives (fake news incorrectly predicted as true). The classification

Table 4. Classification Report of the Lstm Model

	Precision	Recall	F1-Score	Support
Predicted Fake	1.00	0.97	0.98	1721
Predicted True	0.98	1.00	0.99	2144
Accuracy			0.99	3865
Macro Avg	0.99	0.99	0.99	3865
Weighted Avg	0.99	0.99	0.99	3865

report provided a detailed breakdown of these results. For predicted fake news, the precision was 1.00, indicating all predictions for fake news were correct. The recall was 0.97, meaning 97% of actual fake news was correctly identified. The F1-score, which is the harmonic mean of precision and recall, was 0.98, and the support (number of true instances for fake news) was 1721. For predicted true news, the precision was 0.98, indicating a high proportion of true positive predictions among all true news predictions. The recall was 1.00, indicating all actual true news was correctlyidentified, and the F1-score was 0.99 with a support of 2144.

The metrics demonstrated the model's robustness. The accuracy was 0.99, indicating 99% of the total predictions were correct. Both the macro and weighted averages for pre- cision, recall, and F1-score were 0.99, indicating balanced performance across both classes and taking into account the support for each class. These results demonstrate that the LSTM model performs exceptionally well in distinguishing between true and fake news, with high precision and recall values across both classes and minimal misclassifications, making the model highly reliable and effective for this classification task.

7. Conclusion

In this paper, we developed and implemented a fake news detection system utilizing an LSTM model trained on the ISOT Fake News dataset. Our methodology en- compassed comprehensive data preprocessing, including the removal of HTML tags, URLs, hashtags, and other irrele- vant characters, as well as the elimination of stop words and punctuation. This preprocessing step ensured that the textual data was clean and suitable for feature extractionand subsequent modeling.

We employed GloVe embeddings to represent the textual data in a meaningful vector space, capturing semantic rela- tionships between words. The LSTM model was designed

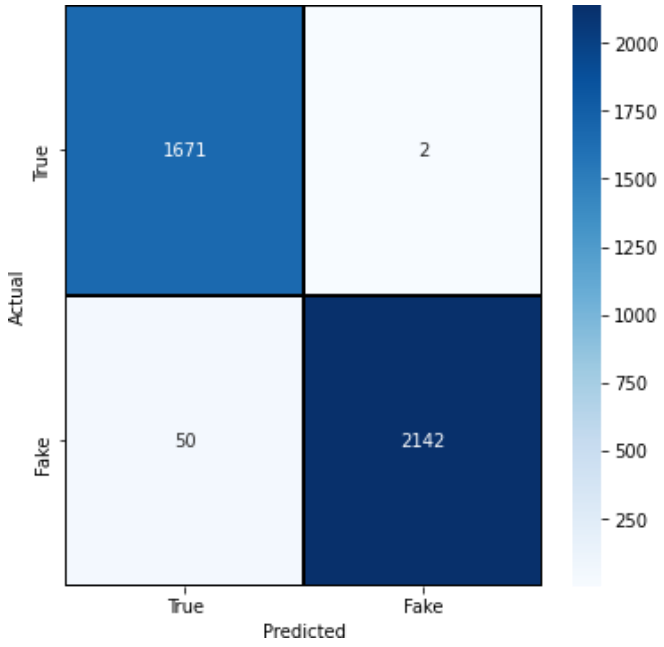

Figure 11. Training & Confusion Matrix

with an embedding layer, followed by two LSTM layers and a fully connected layer, to capture the temporal dependen- cies and contextual information within the text. The model was trained using an Adam optimizer and

evaluated on a separate validation dataset to monitor its performance.

Our experimental results demonstrated that the LSTM model achieved a remarkable accuracy of 99.6% in distin- guishing between true and fake news. The model's precision and recall were nearly perfect, indicating its robustness and effectiveness in identifying fake news articles. The confusion matrix analysis revealed a high true positive rate and a low false positive rate, further validating the model's performance.

Despite these impressive results, it is important to ac- knowledge the inherent biases present in the dataset. The dataset's origin from specific sources, such as Reuters for true news and unreliable websites flagged by Politifact for fake news, may have influenced the model's performance. To enhance the model's real-world applicability, future work will focus on incorporating more diverse data sources and generating adversarial examples to test the model's re- silience.

Additionally, integrating explainability methods such as SHAP values and LIME can provide deeper insights into the model's decision-making process, helping to identify and mitigate potential biases. Improving dataset quality and ensuring a balanced representation of various news sources will be crucial steps in advancing the fake news detection system's robustness and generalizability.

References

1. N. Rai, D. Kumar, N. Kaushik, C. Raj, and A. Ali, "Fake news classification using transformer based enhanced lstm and bert," International Journal of Cognitive Com- puting in Engineering, vol. 3, pp. 98–105, 2022.
2. Zhang, Xichen, and Ali A. Ghorbani. "An overview of online fake news: Characterization, detection, and discussion." Information Processing & Management 57.2 (2020): 102025.
3. Monti, Federico, et al. "Fake news detection on social media using geometric deep learning." arXiv preprint arXiv:1902.06673 (2019).
4. Li, Hang. "Deep learning for natural language processing: advantages and challenges." National Science Review 5.1 (2018): 24-26.
5. Brownlee, Jason. "Deep learning for natural language processing." Machine Learning Mystery, Vermont, Australia 322 (2017).
6. Wang, Dongyang, Junli Su, and Hongbin Yu. "Feature extraction and analysis of natural language processing for deep learning English language." IEEE Access 8 (2020): 46335-46345.
7. Park, Sang-Un. "Analysis of the status of natural language processing technology based on deep learning." The Journal of Bigdata 6.1 (2021): 63-81.
8. Widiastuti, N. I. "Deep learning–now and next in text mining and natural language processing." IOP Conference Series: Materials Science and Engineering. Vol. 407. No. 1. IOP Publishing, 2018.
9. Deng, Li, and Yang Liu. "A joint introduction to natural language processing and to deep learning." Deep learning in natural language processing (2018): 1-22.
10. McMaster, Christopher, et al. "Developing a deep learning natural language processing algorithm for automated reporting of adverse drug reactions." Journal of biomedical informatics 137 (2023): 104265.
11. Aldunate, Ángeles, et al. "Understanding customer satisfaction via deep learning and natural language processing." Expert Systems with Applications 209 (2022): 118309.