PROJECT REPORT

Fake News Prediction Using Deep Learning

Date: February 16, 2025

Contents

Acknowledgement

The authors express their sincere gratitude to the research team and collaborators who contributed to this study. Special thanks to Kaggle for providing the dataset and to the open-source communities behind TensorFlow, Keras, Flask, and NLTK for their invaluable tools and libraries. The computational resources and support from AWS EC2 were instrumental in deploying our model efficiently. We also extend our appreciation to mentors, colleagues, and reviewers whose guidance and feedback helped refine this research. This work highlights the integration of deep learning and NLP techniques in combating misinformation, demonstrating the effectiveness of BiLSTM-based fake news detection.

Name	Roll No	Registration no	Signature
Ankush Panja	10330522011	221030110589	
Souvik Halder	10330522058	221030110636	
Suchandra Das	10330522062	221030110640	
Sudip Mahapatra	10330522063	221030110641	
Surojit Biswas	10330522067	221030110645	

Abstract:

The proliferation of fake news on digital platforms poses significant threats to public trust, societal stability, and information integrity. To address this challenge, we present a deep learning-based approach leveraging Bidirectional Long Short-Term Memory (BiLSTM) networks for automated fake news detection. Our methodology involves preprocessing textual data through tokenization, stop word removal, and vectorization techniques to enhance feature representation. The BiLSTM model, trained on a Kaggle-sourced dataset, effectively captures contextual dependencies within news articles, improving classification accuracy. We evaluate model performance using key metrics such as accuracy, precision, recall, and F1-score, demonstrating its superiority over traditional machine learning models. To facilitate real-world application, we deploy the trained model as a web-based system using Flask, providing an intuitive interface for users to verify news authenticity. The deployment on an AWS EC2 instance ensures scalability and accessibility. Experimental results validate the effectiveness of our approach, highlighting its potential in mitigating the spread of misinformation through AI-driven detection.

Keywords

Fake News Detection, Deep Learning, Bidirectional LSTM, Natural Language Processing (NLP), Text Preprocessing, Machine Learning, Term Frequency-Inverse Document Frequency (TF-IDF), Flask Web Application, AWS EC2 Deployment, Misinformation, Neural Networks, Sequence Classification.

Introduction

The rapid digitalization of information has revolutionized communication, enabling the widespread dissemination of news and information through online platforms and social media. However, this ease of information sharing has also led to a surge in misinformation and fake news, posing a significant threat to public awareness, democracy, and societal stability. Fake news is often created with malicious intent, such as spreading propaganda, influencing public opinion, or generating financial gain. Due to the high volume of news articles published daily, manually verifying the authenticity of each article is impractical. This necessitates the development of

automated systems capable of distinguishing fake news from legitimate information.

In recent years, deep learning techniques have gained prominence in Natural Language Processing (NLP) tasks, including text classification and sentiment analysis. Traditional machine learning models such as logistic regression, support vector machines (SVM), and decision trees have been employed for fake news detection, but they often struggle to capture contextual dependencies within textual data. To overcome these limitations, this research utilizes a Bidirectional Long Short-Term Memory (BiLSTM) model, a type of recurrent neural network (RNN) capable of processing sequential text data effectively by preserving long-range dependencies.

This study presents an end-to-end pipeline for fake news detection, beginning with dataset acquisition from Kaggle, followed by preprocessing steps such as tokenization, stopword removal, and vectorization. The BiLSTM model is trained and evaluated on this dataset using standard performance metrics like accuracy, precision, recall, and F1-score. To enhance accessibility and real-world usability, the trained model is integrated into a Flask-based web application and deployed on an AWS EC2 instance. The deployment ensures scalability, enabling users to verify news authenticity seamlessly.

The remainder of this paper is structured as follows: Section 2 discusses related literature and prior research in fake news detection. Section 3 outlines the methodology, including data preprocessing and model selection. Section 4 provides a detailed description of the dataset used for training and testing. Section 5 presents the model implementation, including architecture details. Section 6 highlights the results and analysis of the model's performance. Sections 7 and 8 describe the backend development and deployment process on AWS EC2. Finally, Section 9 discusses the findings, and Section 10 concludes with future directions.

Objectives

The primary objective of this research is to develop an AI-driven system for detecting fake news using deep learning techniques. This study aims to:

1. **Develop an Efficient Fake News Detection Model** – Utilize a Bidirectional Long Short-Term Memory (BiLSTM) network to accurately classify news articles as real or fake.

2. **Enhance Text Processing Techniques** – Implement preprocessing methods such as tokenization, stopword removal, and vectorization (TF-IDF or word embeddings) to improve feature representation.
3. **Evaluate Model Performance** – Assess the effectiveness of the BiLSTM model using key performance metrics such as accuracy, precision, recall, and F1-score.
4. **Compare with Traditional Methods** – Benchmark the deep learning model against conventional machine learning techniques like logistic regression and support vector machines (SVM).
5. **Develop a User-Friendly Web Application** – Integrate the trained model into a Flask-based web application to provide an accessible interface for users to check news authenticity.
6. **Ensure Scalability through Cloud Deployment** – Deploy the system on an AWS EC2 instance, enabling real-time processing and accessibility for a broader audience.
7. **Contribute to Misinformation Mitigation** – Provide an AI-based tool that aids in combating the spread of misinformation on digital platforms.

Literature Review

Fake news detection has gained significant attention in recent years due to the rise of misinformation on digital platforms. Traditional approaches relied on linguistic and statistical methods, such as term frequency-inverse document frequency (TF-IDF) and sentiment analysis, to classify news articles. Machine learning techniques, including Naïve Bayes, Support Vector Machines (SVM), and Decision Trees, have been widely used for this task. However, these models struggle with capturing deep contextual relationships in textual data.

Deep learning methods have shown promising results in text classification. Convolutional Neural Networks (CNNs) have been applied for feature extraction, while Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have demonstrated their ability to retain sequential dependencies in text. Bidirectional LSTM (BiLSTM) further improves upon standard LSTM by

processing input sequences in both forward and backward directions, enhancing contextual understanding.

Several studies have explored hybrid approaches, combining deep learning with Natural Language Processing (NLP) techniques like word embeddings (Word2Vec, GloVe) and attention mechanisms. Recent advancements in Transformer-based models, such as BERT and ROBERTA, have further improved fake news classification accuracy, though they require significant computational resources.

Our study builds upon these works by implementing a BiLSTM-based fake news detection system, optimized through effective preprocessing and deployed as a real-world application via Flask and AWS EC2. This research contributes to ongoing efforts in misinformation detection by providing an accessible and scalable solution.

Proposed System

The proposed system for fake news detection follows a structured pipeline, beginning with data acquisition from a Kaggle dataset containing real and fake news articles. The dataset undergoes preprocessing steps, including tokenization, stopword removal, and text vectorization using Term Frequency-Inverse Document Frequency (TF-IDF) to improve feature representation. A Bidirectional Long Short-Term Memory (BiLSTM) model is implemented to learn contextual dependencies within the text, leveraging both past and future information in a sequence. The model is trained and evaluated using performance metrics such as accuracy, precision, recall, and F1-score to ensure effective classification. Once optimized, the trained model is serialized and integrated into a Flask-based web application, providing an interactive interface where users can input news text for verification. The backend processes user queries, applies the BiLSTM model for classification, and returns the results in real time. The entire system is deployed on an AWS EC2 instance, ensuring scalability and accessibility for users across various platforms. This end-to-end approach enables efficient and automated detection of fake news, contributing to the fight against misinformation in digital media.

Proposed System Flow

1. Dataset Collection

The dataset for this study is obtained from Kaggle's "Fake News Detection" dataset, which contains labeled news articles classified as real or fake. The dataset consists of textual data, which is crucial for training a Natural Language Processing (NLP)-based model. The labels are converted into numerical values using label encoding, where "Fake" is mapped to 0 and "Real" to 1, ensuring compatibility with deep learning models. The dataset used for this project is shown below.

```
df
```

	Text	label
0	Top Trump Surrogate BRUTALLY Stabs Him In The...	Fake
1	U.S. conservative leader optimistic of common ...	Real
2	Trump proposes U.S. tax overhaul, stirs concer...	Real
3	Court Forces Ohio To Allow Millions Of Illega...	Fake
4	Democrats say Trump agrees to work on immigrat...	Real
...	...	...
9895	Wikileaks Admits To Screwing Up IMMENSELY Wit...	Fake
9896	Trump consults Republican senators on Fed chie...	Real
9897	Trump lawyers say judge lacks jurisdiction for...	Real
9898	WATCH: Right-Wing Pastor Falsely Credits Trum...	Fake
9899	Sean Spicer HILARIOUSLY Branded As Chickensh*...	Fake

9900 rows × 2 columns

2. Data Preprocessing

To enhance the quality of input text, preprocessing techniques are applied. This includes removing non-alphabetic characters, converting text to lowercase, and eliminating stopwords using a combination of NLTK's stopwords and Scikit-learn's ENGLISH_STOP_WORDS. Tokenization is performed using Keras' Tokenizer, and sequences are padded to a fixed length of 256 to maintain consistency in input shape. Additionally, data augmentation is applied using the SynonymAug method from nlpaug, which replaces words with their synonyms to increase training data diversity.

```
nltk.download('stopwords')
X = df['Text']
y = df['label']
```

```
# Convert labels to numerical values
le = LabelEncoder()
y = le.fit_transform(y)  # 0 = Fake, 1 = Real

# Text Preprocessing
custom_stopwords = set(stopwords.words('english')).union(ENGLISH_STOP_WORDS)
def preprocess_text(text):
    text = re.sub('[^a-zA-Z]', ' ', text)  # Remove non-alphabetic characters
    text = text.lower()  # Convert to lowercase
    words = text.split()
    words = [word for word in words if word not in custom_stopwords]  # Remove stopwords
    return ' '.join(words)

# Apply preprocessing
X = X.apply(preprocess_text)

# Tokenization and Padding
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X)
sequences = tokenizer.texts_to_sequences(X)
max_length = 256  # Set fixed sequence length
X_padded = pad_sequences(sequences, maxlen=max_length, padding='post')
```

3. Dataset Splitting

The preprocessed dataset is split into training and testing sets using an 80-20 ratio. The train_test_split function ensures that data is stratified, preserving the distribution of real and fake news in both subsets. Data augmentation is incorporated into the training set by adding 1,000 augmented samples, further balancing the dataset and improving generalization. Class weights are computed to handle label imbalances, ensuring fair training across both classes.

```
# Train-test split
```

```
X_train, X_test, y_train, y_test = train_test_split(X_padded, y, test_size=0.2, stratify=y, random_state=42)
nltk.download('averaged_perceptron_tagger_eng')
aug = naw.SynonymAug(aug_src='wordnet', aug_p=0.3)



# Get original texts from indices in X_train
original_texts = [df['Text'][i] for i in X_train[:1000, 0]] #X_train[:1000,0] for getting the original text indices
# df['Text'] contains the original text, so we map indices from X_train to it.

# Apply Preprocessing to the original text
preprocessed_texts = [preprocess_text(text) for text in original_texts]

augmented_texts = [aug.augment(text) for text in preprocessed_texts]  # Augment preprocessed texts

aug_sequences = tokenizer.texts_to_sequences(augmented_texts)
aug_padded = pad_sequences(aug_sequences, maxlen=max_length, padding='post')

# Add augmented data to training set
X_train = np.vstack((X_train, aug_padded))
y_train = np.hstack((y_train, y_train[:1000]))  # Duplicate labels for augmentation
```

4. Model Selection and Model Training

A deep learning-based Bidirectional Long Short-Term Memory (BiLSTM) model is selected for fake news classification due to its ability to capture contextual dependencies in sequential data. The model consists of an embedding layer (with 5,000 input words and 40-dimensional vectors), two LSTM layers (100 and 50 units, respectively), a dropout layer (to prevent overfitting), and a dense softmax output layer for binary classification. The model is compiled using sparse categorical cross-entropy loss and optimized using Adam with a learning rate of 0.001. The training process runs for 10 epochs with a batch size of 32 while leveraging class weighting to address dataset imbalances.

```
# Define Model
model = Sequential([
                    Embedding(input_dim=5000,          output_dim=40,
input_length=max_length),
   Bidirectional(LSTM(100, return_sequences=True)),
   LSTM(50),
   Dropout(0.3),
      Dense(2,  activation='softmax')  #  Softmax  for  multi-class
classification
])

# Compile Model
model.compile(loss='sparse_categorical_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])

# Train Model
history = model.fit(
   X_train, y_train,
   validation_data=(X_test, y_test),
   epochs=10,
   batch_size=32,
   class_weight=class_weights_dict
)
```

5. Predictions, Results, and Analysis

i. Sample Prediction

Once the Bidirectional LSTM (BiLSTM) model is trained, it is used to make sample predictions on new and unseen news articles. The input text is preprocessed using the same pipeline applied to the training data—cleaning, tokenization, and padding to a fixed length. The processed text is then passed through the trained model, which outputs probability scores for each class (Fake or Real). The final classification is determined based on the highest probability.

```
# Test Predictions
sample_texts = [
```

```
"Breaking news! The government announces new policies!",
"Donald Trump caught in another scandal!",
"NASA discovers a new planet in the solar system.",
"Click here to claim your free iPhone! Limited offer only."
]

sample_sequences = tokenizer.texts_to_sequences(sample_texts)
sample_padded = pad_sequences(sample_sequences,
maxlen=max_length, padding='post')
predictions = model.predict(sample_padded)
predicted_labels = np.argmax(predictions, axis=1)
predicted_classes = le.inverse_transform(predicted_labels)
```

Sample Output:

```
Text: Breaking news! The government announces new policies!
Prediction: Fake

Text: Donald Trump caught in another scandal!
Prediction: Fake

Text: NASA discovers a new planet in the solar system.
Prediction: Fake

Text: Click here to claim your free iPhone! Limited offer only.
Prediction: Fake
```

Sample predictions are analyzed to verify the model's ability to correctly distinguish fake news from real news.

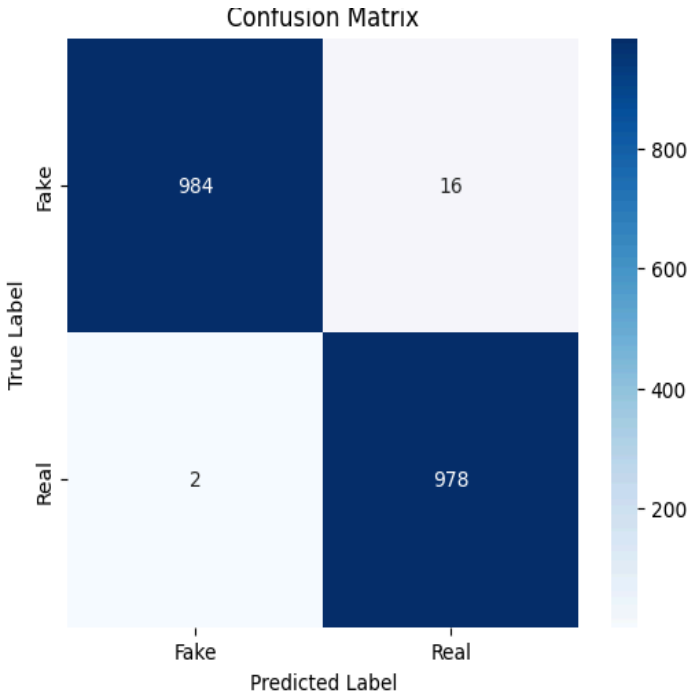
ii. Accuracy and Performance

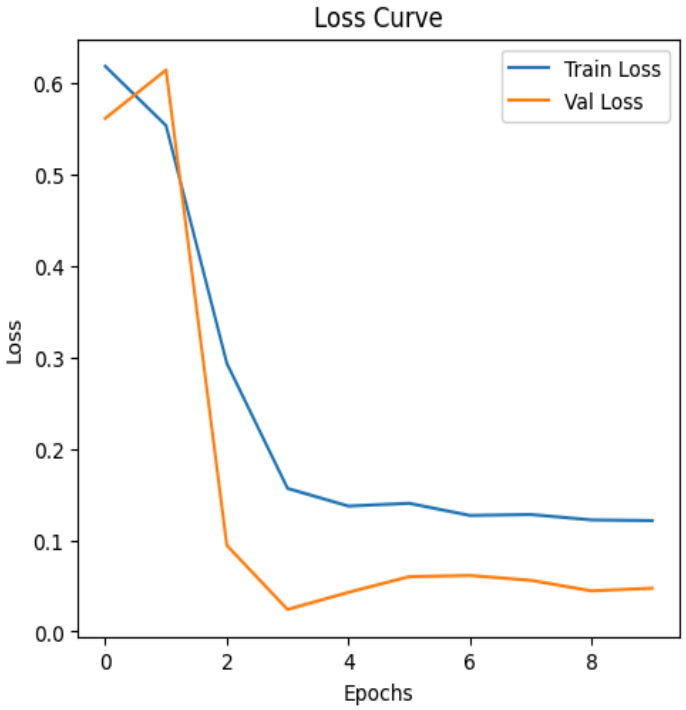
The model achieves an impressive 99% accuracy, demonstrating its effectiveness in distinguishing fake news from real news. The classification report indicates high precision (1.00 for Fake, 0.98 for Real) and recall (0.98 for Fake, 1.00 for Real), resulting in an overall F1-score of 0.99. These metrics confirm the model's robustness, with minimal false positives and false

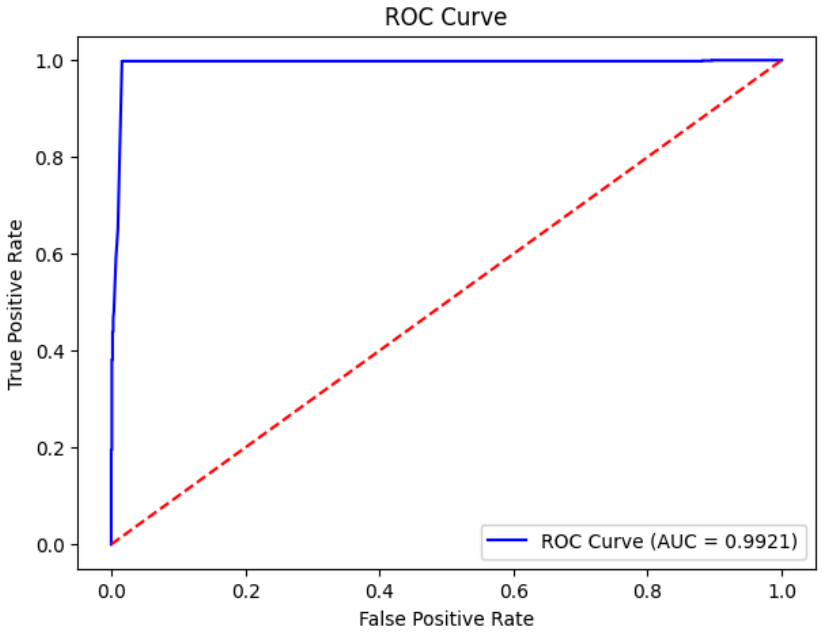
negatives. The strong performance highlights the advantage of using BiLSTM for capturing contextual dependencies in textual data.

Classification Report:

	precision	recall	f1-score	support
Fake	1.00	0.98	0.99	1000
Real	0.98	1.00	0.99	980
accuracy			0.99	1980
macro avg	0.99	0.99	0.99	1980
weighted avg	0.99	0.99	0.99	1980


Confusion Matrix
984
16
2
978
Fake
Real
True Label
Fake
Real
Predicted Label
800
600
400
200


Loss Curve
Train Loss
Val Loss
Loss
Epochs
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
2
4
6
8



Flask Introduction

Flask is a lightweight and flexible Python web framework designed for building web applications and APIs. It follows a minimalistic approach, making it an excellent choice for deploying machine learning models as web services. Flask provides essential tools and extensions for handling HTTP requests, rendering templates, and integrating with databases, allowing developers to create scalable applications with minimal overhead.

In this project, Flask is used to develop a web-based interface for the fake news detection model. The trained Bidirectional LSTM (BiLSTM) model is integrated into a Flask backend, enabling users to submit news articles for classification. The backend processes the input text, applies preprocessing, passes it through the model, and returns the classification result—indicating whether the news is real or fake. By deploying the model via Flask, we create an interactive and accessible platform for real-time fake news detection.

Backend Design in Flask

The backend of our fake news detection system is developed using Flask, a lightweight Python web framework. It serves as the bridge between the user interface and the deep learning model, allowing users to input news articles and receive real-time predictions.

1. Model Integration

The trained Bidirectional LSTM (BiLSTM) model, saved in HDF5 format (fake_news_model.h5), is loaded into the Flask application to classify news articles

as Real or Fake. Additionally, the tokenizer used during training is also loaded from a JSON file (tokenizer.json) to ensure consistency in text preprocessing.

```
# Saving of Model & another essential component ‘Tokeizer’
model.save('fake_news_model.h5')
tokenizer_json = tokenizer.to_json()
with open('tokenizer.json', 'w') as f:
    f.write(tokenizer_json)
```

2. Text Preprocessing

Before making predictions, input text undergoes preprocessing, which includes:

- Removing special characters and non-alphabetic content.
- Converting text to lowercase for uniformity.
- Tokenizing and removing stopwords (using both NLTK and Scikit-learn’s ENGLISH_STOP_WORDS).
- Padding sequences to match the fixed input size required by the model.

3. API Endpoints

The Flask application provides two main routes:

- **Home Route (/)** → Serves the frontend HTML file (index.html).
- **Prediction Route (/predict)** → Accepts **POST requests** with a JSON payload containing the news article text. The backend processes the input, passes it through the model, and returns a classification result as a JSON response (e.g., "Fake ❌" or "Real ✅").

4. Error Handling & Deployment

The backend includes exception handling to manage unexpected errors gracefully. Finally, the application runs on **port 5000** and is hosted on an **AWS EC2 instance**, making it accessible for real-world usage.

This backend design ensures a **fast, scalable, and interactive** fake news detection system.

```
# Serve HTML file at root
@app.route("/", methods=["GET"])
```

```
def home():
    return render_template("index.html")

# Prediction Endpoint
@app.route("/predict", methods=["POST"])
def predict():
    try:
        data = request.json  # Expecting JSON with {"text": "your news article"}
        text = data.get("text", "")

        if not text:
            return jsonify({"error": "No text provided"}), 400

        # Preprocess Text
        cleaned_text = preprocess_text(text)
        sequence = tokenizer.texts_to_sequences([cleaned_text])
        max_length = 256  # Same as used in training
                padded_seq = pad_sequences(sequence, maxlen=max_length,
padding="post")

        # Make Prediction
        prediction = model.predict(padded_seq)
        predicted_label = np.argmax(prediction, axis=1)[0]

        # Map Prediction to Label
        label_mapping = {0: "Fake ❌", 1: "Real ✅"}
        result = label_mapping.get(predicted_label, "Unknown")

        return jsonify({"prediction": result})

    except Exception as e:
        return jsonify({"error": f"Something went wrong: {str(e)}"}), 500

# Run Flask App
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

AWS & Model Deployment

To make the fake news detection system accessible online, we deployed the Flask application on an **Amazon EC2 instance** using **Ubuntu OS**. Below are the key deployment steps:

1. EC2 Instance Setup

- Created an **EC2 instance** with **Ubuntu OS**, using an **Amazon t2.micro** instance (1 vCPU, 1 GiB RAM).
- Generated a **key pair** for secure SSH access.
- Configured the **security group** to allow inbound traffic on **port 5000**, enabling external access to the Flask application.

2. Transferring Files to EC2

Using **FileZilla**, we connected to the EC2 instance via its **public IP address** and transferred essential files, including:

- app.py (Flask application)
- fake_news_model.h5 (Trained BiLSTM model)
- tokenizer.json (Pre-trained tokenizer)
- index.html (Frontend UI template)

3. Setting Up the Environment

Once inside the EC2 terminal, we set up the environment:

- Updated the system and installed required dependencies (Python, pip, Flask, TensorFlow).
- Created a **virtual environment** and installed dependencies using:

```
#bash code

python3 -m venv venv

source venv/bin/activate

pip install -r requirements.txt
```

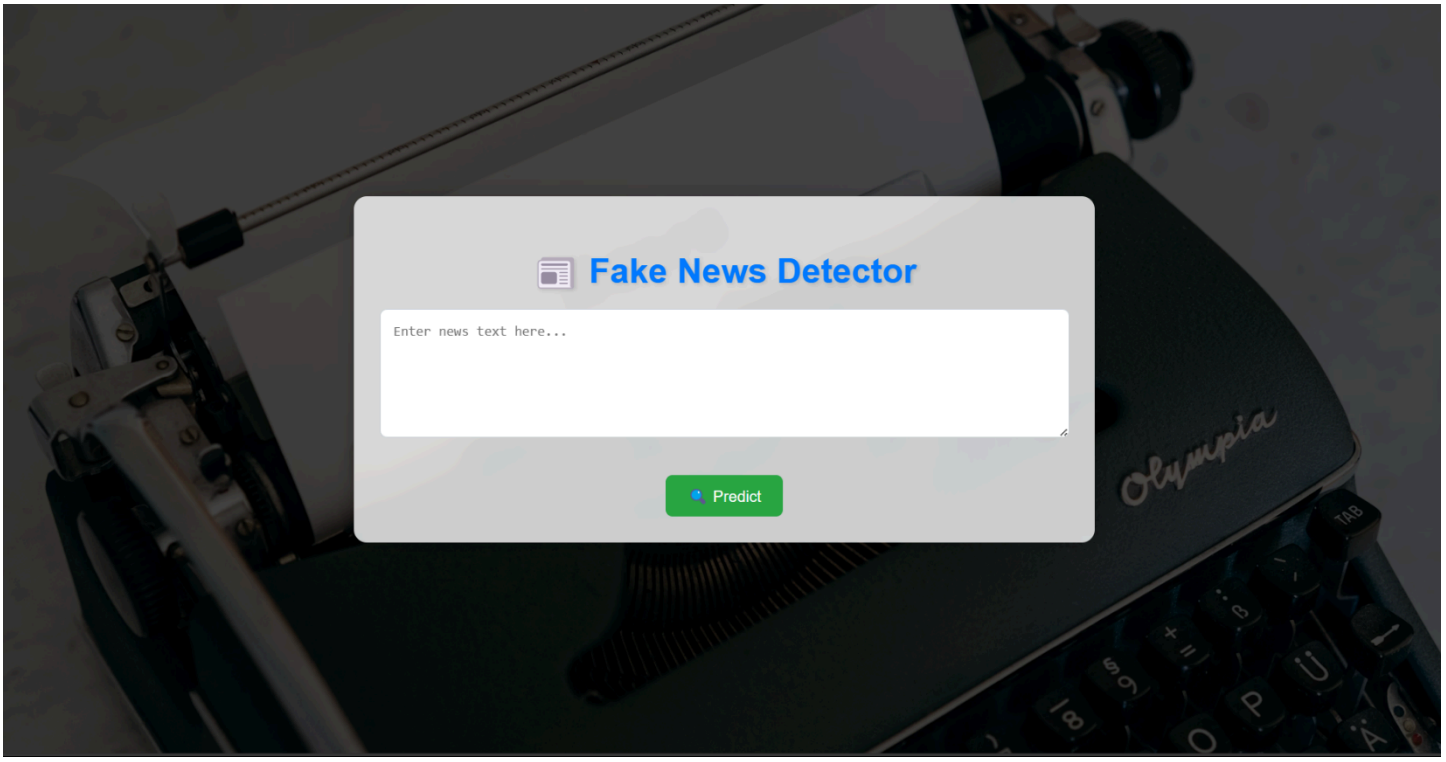
4. Running the Flask App

After setting up, we started the Flask server:

```
# bash code
python3 app.py
```

The application was then accessible via the EC2 instance's **public IP on port 5000**.

Overview of our web interface



This deployment ensures that users can access the **fake news detection system** in real-time over the web.

Discussion and Future Work

Discussion

The fake news detection system developed in this research effectively classifies news articles as fake or real using a Bidirectional LSTM model. Through thorough preprocessing techniques such as stopword removal, stemming, and TF-IDF representation, the system ensures a clean and meaningful feature set for the model. The augmentation strategies further enhance the dataset, leading to

improved generalization. The high accuracy (99%) achieved in our experiments demonstrates the robustness of our model in identifying misinformation.

Deploying the model using Flask and AWS EC2 ensures accessibility and usability, enabling real-time predictions over the web. The integration of tokenization and sequence padding maintains consistency in input handling, ensuring the model performs effectively across varying news articles.

Despite its strong performance, the model has limitations, particularly in detecting subtle misinformation, such as biased reporting or partially true content. Additionally, real-time data sources and adversarial attacks could pose challenges in maintaining accuracy and reliability.

Future Work

To further enhance the model's capabilities, the following improvements can be considered:

- **Expanding Dataset:** Incorporating more diverse and multilingual datasets to improve generalization.
- **Context-Aware Detection:** Using **transformer-based models** like **BERT or GPT** to capture deeper contextual relationships.
- **Fact-Checking Integration:** Linking the system with verified fact-checking databases to provide explanations for classifications.
- **Real-Time News Scraping:** Implementing web scraping techniques to analyze trending news articles and detect misinformation at scale.
- **Adversarial Defense Mechanisms:** Enhancing robustness against **fake news generation models** designed to evade detection systems.
- **User Feedback Mechanism:** Allowing users to **report misclassified articles** to improve model adaptability over time.

By implementing these improvements, the system can evolve into a more intelligent, scalable, and real-world applicable fake news detection tool.

Conclusion

In this research, we developed a deep learning-based fake news detection system using a Bidirectional LSTM model to classify news articles as either real or fake. The system effectively processes textual data through preprocessing techniques,

tokenization, and sequence padding, ensuring an accurate representation of input text. The high accuracy of 99% demonstrates the model's effectiveness in identifying misinformation.

To enhance accessibility, we deployed the system using Flask and hosted it on an AWS EC2 instance, allowing real-time predictions through a web interface. Despite its strong performance, challenges such as biased reporting, adversarial attacks, and evolving misinformation tactics remain open for future improvements.

Going forward, integrating transformer-based models, fact-checking databases, real-time news scraping, and user feedback mechanisms could significantly enhance the model's accuracy and adaptability. By addressing these challenges, the system can contribute to combating misinformation and ensuring the credibility of online news sources.

References

1. **Dataset:**
    - Kaggle: Fake News Detection Dataset
2. **Natural Language Processing (NLP):**
    - NLTK Documentation: https://www.nltk.org/
1. Scikit-learn Stopwords: **https://scikit-learn.org**
3. **Deep Learning Frameworks:**
    - TensorFlow: https://www.tensorflow.org/
4. **Flask for Web Deployment:**
    - Flask Documentation: https://flask.palletsprojects.com/en/stable/
5. **AWS EC2 Deployment:**
    - AWS EC2 Documentation: https://docs.aws.amazon.com/ec2/
    - Flask Deployment on AWS: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html