NewsGuard AI

Real-Time News Verification, Recommendation, and Summarization System

CREATED BY:
SINAN SAMAH
TAHA NAGDAWALA

Table of Contents

Abstract

The spread of fake news across digital platforms has become a major global concern, affecting public opinion, political stability, and access to trustworthy information. To address this issue, this project presents a comprehensive AI system designed to automatically classify news articles as real or fake. This report details the development of NewsGuard AI, a comprehensive system designed to combat fake news through real-time verification, credible article recommendation, and concise summarization. Leveraging a fine-tuned LSTM (Long Short-Term Memory) model for highly accurate fake news classification, NewsGuard AI integrates with the NewsAPI to retrieve relevant, authentic articles and utilizes Hugging Face AI models for on-demand summarization of recommended content. For enhanced transparency and user trust, the system incorporates LIME (Local Interpretable Model-agnostic Explanations) to provide insights into model predictions. The project is deployed as an interactive web application using Streamlit, facilitating broad accessibility, and extensively tested using Streamlit and Gradio(used for initial testing) for rapid prototyping and demonstration. The comprehensive description of each subsystem's development, workflow, and utility demonstrates a structured approach to building a reliable fake news detection solution. The report also discusses key limitations of the current system, such as dataset dependency and lack of multilingual support, and proposes several future enhancements.

Introduction

In an age where misinformation is rapidly disseminated across digital platforms, the ability to detect fake news has become increasingly important. This report presents the design, implementation, and evaluation of a deep learning-based system for fake news detection, developed as part of an applied artificial intelligence project. The system is built using a CuDNN-accelerated Bidirectional LSTM model enhanced with an attention mechanism and hyperparameter tuning via Bayesian optimization.

The model is trained on the WELFake dataset and incorporates a comprehensive preprocessing pipeline to ensure the integrity and quality of input text. Key components of the system include entity extraction, summarization, explainability through LIME, and retrieval of related articles for further verification. This report documents the technical design, architectural choices, comparative evaluation with baseline models, and challenges encountered during development.

Domain of the AI System

The NewsGuard AI system operates within the critical domain of digital news consumption and misinformation detection. Its primary objective is to combat the pervasive issue of fake news and restore public trust in online media. This encompasses analyzing news articles across diverse platforms, including social media, traditional news websites, and various digital publications.

The system's focus encompasses a broad range of news categories, including but not limited to:

- Politics: Both domestic (U.S.) and international political developments, policies, and elections.
- Social Issues: Topics related to societal trends, cultural shifts, and community concerns.
- Economic Policy: News concerning financial markets, government economic decisions, and business trends.
- Legal Matters: Reporting on court cases, legislation, and legal precedents.
- Public Safety: Information related to emergencies, crime, and public health.

NewsGuard AI is primarily designed to serve researchers and data scientists engaged in the study of misinformation, natural language processing, and AI interpretability. For these users, the system offers a robust platform for analyzing news veracity and, critically, for delving into the underlying mechanisms of AI-driven content assessment through its LIME-powered explanations.

Beyond this core audience, NewsGuard AI also offers significant value to educators, students, and the general public seeking reliable information and a deeper understanding of digital media literacy. The system's relevance is critical due to the unprecedented volume of misinformation across digital platforms, consistently outpacing traditional human fact-checking efforts. By providing an intelligent solution with transparent decision-making, NewsGuard AI aims not only to restore confidence in digital news sources but also to empower a more informed and analytically capable public discourse.

Requirements and Specifications of the System

This section details the comprehensive requirements and specifications for the NewsGuard AI system, encompassing its objectives, the problem formulation, and the functional and non-functional aspects of its design.

Objectives

The AI-powered fake news detection, recommendation, and summarization system shall:

- **FND-001 (Classification Accuracy):** THe WelFake dataset includes English-language news articles from U.S. and international politics, social issues, economic policy, legal matters, and public safety, which are pulled from social media or news websites, as real or fake. The system, utilizing the CuDNNLSTM model, has achieved the following performance metrics on the test dataset:
    - **Test Accuracy: 93.01%**
        - Out of all articles tested, 93 out of every 100 were correctly classified as either real or fake. This high accuracy indicates strong overall model performance.
    - **Test AUC Score:** 0.9802
        - With an AUC close to 1, the model is highly effective at distinguishing between fake and real news. It means that in 98% of cases, the model ranks a randomly chosen real article higher than a fake one, showing excellent discriminatory ability.
    - **Test Loss:** 0.3478
        - This relatively low loss value suggests that the model's predicted probabilities are close to the actual labels. It reflects good generalization
    - **Precision:** 0.9283
        - Out of all articles the model labeled as fake, around 93% were truly fake. This helps minimize false alarms, ensuring real news is not wrongly flagged as fake.
    - **Recall:** 0.9456
        - The model correctly identified about 95% of all fake news articles. This high recall is crucial because it means the system successfully catches most misinformation.
    - **F1-Score:** 0.9369
        - The model balances precision and recall, indicating that the model handles the trade-off between missing fake articles and wrongly flagging real ones very well. A score of 0.9369 confirms overall robustness.
    - **Specificity:** 0.9113

  - About 91% of real news articles were correctly identified as real. This means the system preserves trust in legitimate content while still being vigilant about misinformation.

Together, these numbers confirm that the system performs consistently well in identifying fake news while minimizing errors—both false positives and false negatives.

- **FND-002 (Confidence Scoring):** Every classification shall have a confidence score from 0% to 100%, giving users a quantifiable measure of the system's certainty. Various calibration methods were tested and applied to ensure that the model does not get over- or under-confident. The confidence scores will be categorized into high, uncertain, and low. Upon uncertain mid range scores, the system will encourage the user to research further for a concrete outcome.
- **FND-003 (Scalability for Concurrent Users):** The system shall support up to five concurrent users without significant degradation in performance (e.g., response time increase of more than 20%). This threshold is set to ensure a practical level of accessibility for a university student project, balancing resource constraints with usability.
- **FND-004 (User-Friendly Explanations):** Results, including classifications, confidence scores, and summaries of recommended articles, shall be explained in simple, non-technical language understandable by a general audience, including individuals with a high school education level.
- **FND-005 (User Privacy):** User privacy shall be protected by not storing any articles they submit or their interaction history, in compliance with general data protection principles. This ensures that personal data is not retained beyond the immediate processing needs.
- **FND-006 (Anonymous Interface):** Provide a user-friendly anonymous interface with reasonable performance, allowing users to interact with the system without requiring personal accounts or logins.
- **FND-007 (Model Interpretability):** The system shall provide local explanations for its classification decisions, highlighting the words or phrases that most strongly influenced the LSTM model's prediction for a given news article. This interpretability feature, powered by LIME, aims to increase user trust and understanding of the AI's reasoning (Ribeiro et al., 2016).

Dataset

For the development and evaluation of Artificial Intelligence (AI) models aimed at detecting fake news, the WELFake dataset has been selected as the primary data source. This dataset is widely recognized within the academic and research communities for its comprehensive nature and suitability for text-based fake news classification tasks.

The WELFake dataset comprises 72,134 news articles, meticulously labeled as either "real" (0) or "fake" (1). This balanced distribution, with 35,028 real news articles and 37,106 fake news articles, is crucial for preventing bias in model training and ensuring robust performance across both classes. Each entry within the dataset typically includes the following critical features for AI processing:

- **Serial Number:** A unique identifier for each news article, facilitating data management and traceability.
- **Title:** The headline of the news article, which often contains key information or deceptive cues that AI models can leverage.
- **Text:** The full body content of the news article, serving as the primary input for natural language processing (NLP) and deep learning models to extract linguistic patterns, stylistic inconsistencies, and factual discrepancies.
- **Label:** The binary classification (0 for fake, 1 for real), providing the ground truth for supervised learning algorithms.

A significant advantage of the WELFake dataset lies in its aggregation from four prominent and diverse news sources: Kaggle, McIntire, Reuters, and BuzzFeed Political. This multi-source aggregation strategy is paramount for developing AI models that do not overfit the characteristics of a single dataset. By incorporating a broader spectrum of writing styles, topics, and patterns associated with both authentic and fabricated news, WELFake enables the training of a more generalized fake news detection system. The dataset's focus on textual content allows for the direct application of advanced NLP techniques, including word embeddings, sequence modeling, and attention mechanisms, without reliance on external metadata, thereby streamlining the development of AI-driven textual analysis solutions for fake news identification.

Problem Domain and Formulation

Problem Domain: The core problem is fake news's rapid dissemination and impact. False information travels faster and broader than factual news, often leveraging emotional language and biased viewpoints to appear credible (Vosoughi et al., 2018). Human fact-checking alone cannot keep pace with the sheer volume of online content. This leads to public confusion, misinformed decisions, and a decline in trust in legitimate news sources, particularly during critical events.

NewsGuard AI addresses this by:

1. **Classification**: When given a news article (title and text), determine its class(Real/fake). This is framed as a binary classification problem.
    - Input: Textual content of a news article.
    - Output: A binary label (Real or Fake) and a confidence score.
2. **Recommendation and Summarization**: Given a topic (derived from a classified article or user query), retrieve relevant, authentic news articles and provide concise summaries. This involves:
    - Information Retrieval: Querying external news APIs (NewsAPI) based on keywords or topics extracted from the input article.
    - Content Filtering: Selecting articles from reputable sources.
    - Summarization: Generating a brief, informative summary of the retrieved articles using Hugging Face facebook/bart-large-cnn model.
3. **Interpretability:** For a given classification, identify and visualize the key textual features (words/phrases) that contributed most significantly to the model's prediction. This is formulated as a feature importance explanation problem, addressed using LIME.

This multi-faceted approach moves beyond simple detection, aiming to provide a comprehensive solution that identifies misinformation, actively guides users towards verified information, and helps them quickly grasp its essence.

Why the Developed AI System Can Help

The NewsGuard AI system offers several compelling advantages in the fight against misinformation:

- **Advantages over other models: W**hile models like Perplexity or ChatGPT are powerful general-purpose language tools, they are not optimized for fake news detection. They provide subjective, conversational responses rather than structured classifications like "REAL" or "FAKE," and they lack dedicated training on labeled misinformation datasets. Our system, by contrast, is purpose-built for this task, offering high-accuracy predictions, clear model explanations via LIME, and fact-based external validation for real news.

  Compared to other misinformation-trained models, our system offers greater transparency, external validation, and ongoing adaptability. It uses LIME for explainability, retrieves verified sources for real news, and avoids amplifying fake content. Moreover, our system uses aggressive preprocessing to reduce noise and improve model focus on meaningful content.

- **Real-time Verification:** Unlike manual fact-checking, which is often retrospective and slow, NewsGuard AI provides near real-time classification. This speed is critical in preventing the rapid spread of fake news, especially during breaking events.

- **Efficient and Accurate Classification with Deep Learning:** By leveraging a fine-tuned LSTM model, the system efficiently processes textual data to identify subtle linguistic patterns, stylistic anomalies, and deceptive rhetoric often employed in fake news. LSTM's computational lightness compared to more complex models like BERT allows for faster inference times and more efficient deployment, which is crucial for real-time applications and within the resource constraints of a university project. The model's demonstrated accuracy (as detailed in FND-001) ensures reliable performance.

- **Proactive Information Guidance:** The system does not just label content as fake; it actively provides verified alternatives. Integrating with NewsAPI fetches authentic articles on the same topic, allowing users to cross-reference information and gain a balanced perspective. This fosters critical thinking and media literacy.

- **Efficient Information Consumption (Summarization with Hugging Face AI):** The integration of Hugging Face Facebook BART model for summarization addresses the information overload prevalent online. Users can quickly grasp the main points of recommended articles without having to read lengthy texts, making it easier to consume and compare multiple credible sources. Hugging Face provides access to a vast ecosystem of pre-trained and fine-tuned models, offering flexibility and state-of-the-art performance for text summarization tasks.

- **Enhanced Trust through Interpretability (LIME):** A significant advantage of NewsGuard AI is its integration of LIME. This feature allows the system to explain *why* it made a particular classification, highlighting the specific words or phrases in the news article that most influenced its decision. This transparency is vital for building user trust, especially when dealing with sensitive topics like fake news. Understanding the model's reasoning helps users develop their critical discernment skills.

- **Accessible Deployment (Streamlit) and Streamlined Testing (Gradio):** The choice of Streamlit for deployment provides a user-friendly and easily accessible web interface for the system. This simplifies interaction for non-technical stakeholders and end-users, making the powerful AI backend readily available. For internal testing and rapid iteration during development, Gradio provides a quick way to create shareable demos of the LSTM model's performance and LIME explanations, streamlining the development and evaluation process. Deploying the system locally with Streamlit allows quick, interactive testing of the fake news detector in a user-friendly web interface. It enables easy input of articles, real-time classification, and visualization of explanations like LIME outputs. This setup helped us to efficiently evaluate model behavior and debug issues.

AI System Design

System Engineering Lifecycle

The development of NewsGuard AI adhered to the systematic and structured methodology of the NASA System Engineering Product Life Cycle (NASA, 2018). This framework provides a disciplined approach to project management, ensuring that all phases, from initial concept to final testing, are conducted rigorously. This approach was instrumental in guiding technical decisions, managing project scope, and ensuring that the final product aligns with its stated objectives.

Pre-Phase A: Concept Studies

Tasks: This initial phase focused on identifying the core problem—namely, the widespread societal impact of fake news and misinformation in the digital age. A set of project objectives (FND-001 to FND-007) was defined to establish the intended outcomes, system capabilities, and success criteria. These included achieving high classification accuracy, supporting interpretability, ensuring data fairness, and preparing for future scalability. A feasibility review of AI-based solutions was conducted, assessing multiple approaches such as rule-based systems, traditional machine learning classifiers (SVM), deep learning methods (LSTM), and pretrained models (BERT).

Initial research was also performed to identify suitable datasets. The WELFake dataset was shortlisted for its size, balance between real and fake samples, and usage in peer-reviewed research.

Outcome: A clearly defined project concept with realistic technical and functional goals. The team gained a deep understanding of the problem domain, existing gaps in the field, and what the system would need to achieve to be useful and reliable.

Phase A: Concept & Technology Development

Tasks: This phase transitioned the project from theoretical planning to early experimentation. Multiple AI architectures were explored through prototyping, including a Bidirectional LSTM with an attention mechanism and a BERT-based classifier. These prototypes were evaluated for feasibility, complexity, interpretability, and performance. The WELFake dataset was selected as the main data source due to its credibility and richness , as cited in related studies (Verma et al., 2021).

During this phase, essential tools and frameworks were selected: TensorFlow, NLTK, in-built python libraries and Keras for model implementation, Scikit-learn for evaluation and preprocessing tasks, and Google Colab as the cloud-based development environment.

Outcome: This phase validated the core AI technologies and provided a working basis for comparing different modeling strategies. Based on interpretability needs and computational efficiency, the LSTM-based model with attention was chosen for final development.

Phase B: Preliminary Design

Tasks: In this phase, the system's architecture was fully designed and broken down into modular components. The full processing pipeline was defined, including:

- **Data Preprocessing**: Tokenization, padding, stopword removal. A custom list was used to remove words that were causing data leakage and bias. These words would repeat in a specific class and carry high influence, making the model unreliable.

- **Feature Engineering**: Extraction of auxiliary content-based features such as sentiment scores, punctuation count, capital letter usage, and word count. The features were identified by conducting thorough exploratory data analysis.

- **Model Training**: Designing dual input streams to process text sequences and auxiliary features in parallel.

- **Evaluation Pipeline**: Defining validation and test procedures, as well as key performance metrics (accuracy, F1-score, AUC-ROC, calibration error).

The data flow was carefully mapped out—from raw article input to final fake/real classification—ensuring that all necessary inputs were processed correctly and in sync. The design also accounted for interpretability and extensibility (e.g., allowing future integration of multimedia content or language support).

Outcome: A complete architectural blueprint was produced, clearly defining system modules, their interactions, and the expected outputs. This formed the foundation for full-scale implementation in the next phase.

Phase C: Detailed Design

Tasks: This phase involved the complete implementation and optimization of the final model architecture. The chosen design, a Bidirectional LSTM with Attention, was implemented to capture long-term dependencies in news text and highlight key informative words. A secondary input stream for auxiliary features was built using fully connected dense layers.
To ensure the best possible model configuration, Bayesian optimization was used to tune hyperparameters such as:

- Number of LSTM units
- Dropout rates
- Learning rate
- Batch size

Additionally, a reliable pipeline for content-based feature extraction was integrated into the system. Once trained, the model underwent interpretability testing using LIME, which highlighted the words contributing most to the model's predictions. These results provided valuable transparency for both researchers and end-users.

Outcome: A fully implemented and optimized AI model capable of detecting fake news with strong performance and explainable behavior. The design ensured not only accuracy but also interpretability and extensibility.

Phase D: System Assembly, Integration, Test, and Launch

Tasks: In the final development phase, all system components were assembled into a unified and reusable pipeline. The model was evaluated on a held-out test set to assess generalization. In addition to standard performance metrics, a critical focus was placed on probability calibration. The model's confidence outputs were analyzed using reliability diagrams, both before and after applying isotonic regression, which significantly improved the alignment between predicted probabilities and true likelihoods.

A comprehensive interpretability analysis was performed, including:

- Word importance plots for fake and real articles
- Class-level feature analysis
- LIME explanations for individual predictions

All the subsystems including user interface, CuDNNLSTM, calibration analysis,and LIME analysis were integrated and tested with news articles from various sites worldwide and fake news as well.

Outcome: A fully validated AI system, tested across technical and interpretability metrics. The system is ready for deployment and accompanied by a clear roadmap for future updates, integration, and scaling.

Overview of the System

NewsGuard AI is designed as a modular system, comprising several interconnected subsystems that collaborate to deliver its core functionalities. This architecture ensures maintainability, scalability, and the ability to integrate future enhancements.

Core Workflow

1. **User Input:** A user provides a news article via the web interface.
2. **Preprocessing:** Before feeding the text data into the model, it undergoes preprocessing to clean and standardize the input. This process ensures that the input text is cleaned, normalized, and ready for embedding and sequential modeling.
3. **Extra Clues are Extracted**: The system also looks at extra clues from the article beyond just the words. These include: number of words, occurrence of punctuation, capital letters, and tone of the article. These additional clues help the system better understand the writing style, which is often different in fake and real news.
4. **Fake News Classification:** The preprocessed text is fed into the Bidirectional LSTM with Attention to determine its class (Real/Fake) alongside the extra clues. This allows the model to focus on certain important phrases.
5. **Probability (Confidence) Calculation:** The system then calculates how likely it is that the article is fake or real. This probability score is carefully calibrated, meaning it is adjusted to make sure it truly reflects how confident the system is in its prediction.
6. **Interpretability:** If the user wants to understand why the article was classified as fake or real, they can view an explanation generated by the system using LIME. This explanation highlights the words that had the most impact on the prediction.
7. **Entity Recognition:** If the article is classified as "Real", relevant keywords, organization, locations, and short phrases are extracted from the article using spaCy.
8. **News Retrieval:** NewsAPI is queried with these keywords to fetch authentic news articles.This helps users cross-check whether the same news topic appears in other trusted sources
9. **Summarization:** The retrieved authentic articles are summarized using the Hugging Face Facebook BART model. This is especially useful for users who want to quickly understand the core of a long news piece.
10. **Results Presentation:** All results including classification, confidence, LIME explanation, recommended articles, and their summaries are presented to the user in a clear, intuitive format.

Subsystem Breakdown

The NewsGuard AI system is composed of the following key subsystems:

User Interface (UI) Subsystem

The User Interface (UI) Subsystem of the NewsGuard AI application is responsible for facilitating user interaction with the underlying fake news detection engine. It is built using the Streamlit framework, enabling rapid development of an interactive, web-based application with real-time responsiveness.

- **Design & Styling:** The interface adopts a modern dark theme with high visual contrast, leveraging glassmorphism aesthetics and subtle animations to enhance usability. Custom CSS is embedded directly into the application to override default Streamlit styles, allowing for:
    - Gradient backgrounds and component shadows
    - Rounded containers and cards for a premium visual experience
    - Animated loaders and progress bars
    - Font Awesome icons and Google Fonts (Inter)
- **Layout:**The application follows a two-panel layout, consisting of a collapsible sidebar and a main content area, structured as follows:
    - Sidebar: Contains descriptive sections such as About, Features, and Model Accuracy. Moreover, it uses tooltips to provide concise explanations of system capabilities (e.g., LIME explanations, summarization, entity extraction).
    - Main Panel: It serves as the central workspace where users interact with the system. It includes input fields for the article title and full text, followed by an “Analyze Article” button that initiates the fake news detection process. Once submitted, the panel guides the user through several stages: preprocessing the input, displaying the prediction outcome (REAL or FAKE) with confidence scores, and presenting interpretability results using LIME visualizations and highlighted text. The bottom of the main panel is split into 2 tabs, adding to the design element of the page. The first tab is for AI Explanation where for every article it highlights key terms that contributed to the classification. The second tab displays related verified news sources along with AI-generated summaries if the article is deemed real. Throughout the process, progress indicators and status cards provide real-time feedback, ensuring an informative and interactive user experience.
- **Feedback:** The UI provides real-time feedback at each processing stage using animated loaders, status messages, and progress bars. Alerts are generated for:
    - Missing article title or text
    - Short or excessively long article text
    - Missing API keys or model files

Fake News Classification Subsystem (CuDNNLSTM Bidirectional Model)

- **Purpose:** This is the core AI engine responsible for classifying the news articles.
- **AI Technique:** The CuDNNLSTM Bidirectional model is a deep learning architecture that leverages NVIDIA's GPU-accelerated CuDNN library to speed up the training and inference of LSTM layers. It uses Bidirectional LSTM layers, which process text sequences in both forward and backward directions, enabling the model to capture contextual information from both past and future tokens. This bidirectional structure enhances the model's ability to understand complex linguistic dependencies in news articles. The final output is passed through dense layers to classify the article as either real or fake.
- **Dataset**: The project utilized the **WELFake dataset**, a large and balanced corpus containing 72,134 news articles. The dataset is composed of 37,106 "Real" articles (51.4%) and 35,028 "Fake" articles (48.6%), providing a nearly even class distribution ideal for training a binary classifier and minimizing inherent bias(Verma et. al, 2021).
- **Initial Analysis**: The EDA revealed several key characteristics. Missing values were present but minimal, with 0.77% of titles and 0.05% of text bodies being null. Text length analysis showed that real news articles tended to have longer titles and significantly longer text bodies than fake ones. Data quality analysis identified 9,786 duplicate titles and 9,428 duplicate texts, indicating the need for a robust deduplication step in the preprocessing pipeline.
- **Preprocessing Steps:**
    - **Text Cleaning:** Raw news article text (title and body) undergoes rigorous cleaning as implemented in the clean_text function in utils.py. This involves:
        - Remove duplicate articles based on combination of title and text.
        - Converting all text to lowercase.
        - Removing URLs (http[s]?://...) and email addresses (\S+@\S+).
        - Removing HTML tags (<.*?>).
        - Expanding common English contractions (e.g., "won't" to "will not").
        - Aggressive removal of custom stop words which introduce bias into the system. (For example 'reuters', 'cnn')
        - Lemmatization to reduce words to their root form.
        - Removing excessive whitespace and stripping leading/trailing spaces.
        - *Rationale:* These steps are crucial for consistency, reducing noise, and improving the model's ability to generalize during fake news detection. Removing duplicate articles based on the combination of title and text prevents data leakage and redundancy. Lowercasing standardizes the input, allowing the model to treat words like "News" and "news" equally. Eliminating URLs, email addresses, and HTML tags removes irrelevant or potentially misleading content that does not contribute to the semantic understanding of the article. Expanding contractions ensures that words are fully represented for better token matching. Removing biased stop words (like "reuters" or "cnn") prevents the model from learning superficial patterns based on source names. Lemmatization helps unify different inflected forms of words into a common base, improving word matching

and reducing vocabulary size. Lastly, cleaning up whitespace improves the overall text formatting and avoids misalignment during tokenization. Together, these steps help build a cleaner and more robust dataset for accurate and fair predictions.

- **Feature Engineering**: The model's design goes beyond simple text classification by incorporating engineered features. Such as:
  - Title Weighting: the article's title is doubled and prepended to the article text. This gives the title increased weight, acknowledging its often critical role in conveying the core message of an article.
  - Attention Mechanism: The mechanism allows the neural network to focus on specific words/tokens of its input sequence to make a prediction. It consists of Dense, Flatten, and Softmax layers that compute a weight for each timestep. These weights are then used to compute a weighted sum of the LSTM's outputs, effectively allowing the model to focus on the most salient parts of the text when making its decision, enhancing interpretability.
  - Content Based Features: A set of six stylistic features was extracted from the article. This was done to capture raw stylistic signals before normalization. The features are: exclamation mark ratio, question mark ratio, capital letter ratio, average word length, sentence count, and average sentence length. These features provide the model with a parallel stream of information about the article's writing style, making it more robust.
- **Tokenization:** The cleaned and combined text is then tokenized,, which converts words into numerical representations. This involves:
  - Mapping each word to a unique integer ID based on the vocabulary learned during training.
  - Handling out-of-vocabulary (OOV) words.
  - **Bias-Aware Tokenizer**: A fail safe custom BiasAwareTokenizer class was implemented, in case text cleaning fails to remove stop words. This tokenizer, in addition to standard tokenization, removes a predefined list of words that could introduce source-based bias (e.g., 'cnn', 'fox', 'breitbart') or temporal bias (e.g., 'monday', 'january'). By removing these words, the model is forced to make its classification based on the article's content and style, rather than learning simple, brittle rules like "articles from source X are always real".
  - **Vocabulary and Padding**: The tokenizer was configured with a vocabulary size (VOCAB_SIZE) of 20,000, capturing the most frequent and meaningful words in the training corpus. All text sequences were then padded or truncated to a uniform length (MAX_SEQUENCE_LENGTH) of 500 tokens. A 'post' strategy was used for both padding and truncation, meaning padding was added to the end of shorter sequences and tokens were removed from the end of longer ones.

        - *Rationale:* Neural networks require numerical input. Tokenization converts human language into a machine-readable format, and fixed-length sequences are necessary for batch processing and efficient computation within the LSTM architecture. The MAX_SEQUENCE_LENGTH of 500 is a common practice that balances retaining sufficient context with computational efficiency for LSTM models. Post-padding/truncation is chosen for consistency with common NLP practices.
    - **Embedding:** The integer-encoded tokens are converted into dense vector representations (word embeddings). These embeddings capture semantic relationships between words.
        - *Rationale:* Embeddings allow the model to understand the meaning and context of words, rather than just treating them as discrete symbols.
- **CuDNNLSTM Architecture and Parameters:** The LSTM model is loaded from best_lstm_model.h5 and is typically structured with:
    - An Embedding Layer: Maps token IDs to dense vectors.
    - One or more Bidirectional LSTM Layers: Process sequences in both forward and backward directions, capturing long-range dependencies and context from both past and future words. This is particularly effective for understanding nuanced language in news articles.
    - Dense Layers: For classification, mapping the LSTM's output to the final prediction.
    - Output Layer (Sigmoid for binary classification): Produces the probability of the article being "Real" or "Fake".
- **Training:** The model is trained on the WELFake dataset (72,134 articles), split into training, validation, and test sets.
    - **Epochs:** The number of passes over the entire training dataset.
    - **Embedding_dim:** Defines the size of the vector space in which words are represented (i.e., word embedding dimension).
    - **LSTM Units:** Specifies the number of memory cells in the LSTM layer.
    - **Dropout Rate:** Sets the proportion of neurons randomly deactivated during training to reduce overfitting.
    - **Dense Units:** Determines the number of neurons in each dense (fully connected) layer after the LSTM.
    - **Optimizer:** Chooses the algorithm (e.g., Adam, RMSprop) used to update weights and minimize the loss during training.
    - **Number of Dense Layers:** Indicates how many fully connected layers are added after the LSTM layer.
    - **Batch Size:** The number of samples processed before the model's internal parameters are updated.
    - **Learning Rate:** Controls the step size during model optimization.
    - **Optimization:** Standard optimization algorithms (e.g., Adam) are used to minimize the loss function (e.g., binary cross-entropy).

- **Development:** Implemented using deep learning frameworks like TensorFlow. The trained model is saved and loaded for inference within the Streamlit application.

BERT vs LSTM vs CDNN

The selection of a neural network architecture for the fake news classification task within NewsGuard AI involved a careful evaluation of model complexity, computational efficiency, and classification performance. We considered three prominent architectures: Bidirectional Encoder Representations from Transformers (BERT), standard Long Short-Term Memory (LSTM) networks, and the GPU-optimized CuDNNLSTM.

➢ BERT vs. CuDNNLSTM/LSTM:

BERT, as a transformer-based model, is renowned for its ability to capture deep contextual relationships and achieve state-of-the-art accuracy across numerous Natural Language Processing (NLP) tasks. While preliminary experiments indicated that BERT could yield slightly superior accuracy and performance metrics, its computational expense presented a significant challenge. BERT models are considerably larger and demand substantially more GPU memory and processing power for both training and inference. For a university project operating under resource constraints and with a critical requirement for near real-time performance, the marginal gain in accuracy offered by BERT was deemed not advisable due to the disproportionate increase in computational overhead. This decision prioritizes a balance between high performance and practical deployability.

Furthermore, BERT, with its deep transformer architecture and multi-head self-attention layers, captures complex language patterns but behaves like a black box. In contrast, LSTM models when paired with attention mechanisms, are simpler and more transparent, allowing easier interpretation of how word sequences influence predictions. This makes LSTM models more suitable in our use case.

➢ CuDNNLSTM vs. LSTM:

Between the standard LSTM and its GPU-optimized CuDNNLSTM, the latter emerged as the superior choice for NewsGuard AI, offering a balance of modeling capabilities and computational efficiency.

- **GPU-Accelerated Performance:** While standard LSTMs are effective at capturing long-term dependencies in sequential data, their recurrent nature can limit parallelization, leading to slower processing times, especially for longer text sequences. CuDNNLSTM, in contrast, is a highly optimized implementation of the LSTM layer designed to leverage NVIDIA's cuDNN library on compatible GPUs. This fundamental difference translates into significantly faster training and inference speeds compared to standard LSTM layers when executed on a GPU.
- **Enhanced Computational Efficiency:** The optimizations within CuDNNLSTM allow for more efficient utilization of GPU resources, translating into reduced processing time per

epoch and faster overall model convergence. This efficiency is crucial for meeting the system's real-time processing requirements without necessitating high-end, dedicated hardware.

- **Facilitating Hyperparameter Tuning:** The substantial speed improvements offered by CuDNNLSTM directly enable more extensive and rapid hyperparameter tuning. With faster iteration cycles for training and evaluation, developers can explore a broader range of model configurations (number of units, layers, dropout rates) within a given timeframe. This accelerated experimentation is instrumental in identifying an optimal CuDNNLSTM model configuration that maximizes classification performance for the specific fake news detection dataset, leading to a more robust and fine-tuned solution.

In conclusion, while BERT offers peak accuracy, its computational demands are excessive for this project's scope. Between the recurrent architecture, CuDNNLSTM stands out as the optimal choice. Its GPU-accelerated computational efficiency allows for more agile development, comprehensive model optimization through hyperparameter tuning, and ultimately, a highly performant and deployable system that effectively balances high accuracy with practical resource constraints.

Evaluation Discussion

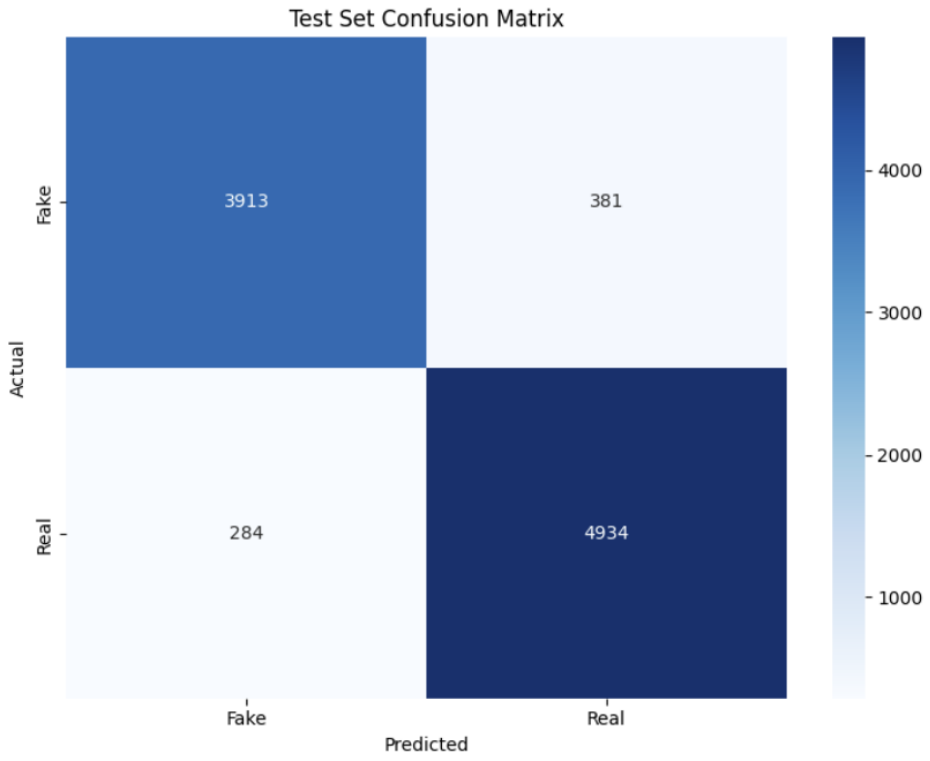

Figure 1

Actual	**Real**	**91.13%**	**8.87%**
	Fake	**5.44%**	**94.56%**
		Real	**Fake**
		Predicted	

	Precision	**Recall**	**F1-Score**
Fake News	**0.9323**	**0.9113**	**0.9217**
Real News	**0.9283**	**0.9456**	**0.9369**
Accuracy	**93.01%**		

These metrics help us understand how well the model is performing in different aspects. The model's overall performance is captured by Accuracy, which tells us the percentage of all news articles that were correctly classified as either fake or real. An accuracy of 95.40% means that the model correctly identified the nature of the news (fake or real) in more than 95 out of every 100 cases. We also look at Precision and Recall for each category. Precision, for example, tells us that when the model labels a news article as "Fake News," it is correct 95.52% of the time, meaning it rarely misidentifies real news as fake. Recall, on the other hand, indicates that the model successfully catches 96.11% of all actual "Fake News" articles, showing its effectiveness at detecting fake content. The F1-Score combines these two measures into a single value, providing a balanced view of the model's performance for each type of news. High F1-Scores (95.82% for Fake News and 94.90% for Real News) demonstrate that the model is consistently strong at both correctly identifying news types and avoiding errors for both fake and real news categories.

Probability Calibration Subsystem

The goal is to make sure that when the model predicts a probability, for example that a news article is 80% likely to be fake it is truly fake around 80% of the time. Without calibration, the model might give high confidence scores like 95% even when it's wrong. This causes legitimate news to be flagged as fake or missing fake news due to misleading confidence levels.

Articles that fall in a gray area (i.e. <70%) can be reviewed by journalists or researchers. The calibration scores allow them to prioritize which articles need to be further investigated. Moreover, the scores help to uncover the blackbox nature of the model, it adds interpretability and combining it with LIME opens the scope for a better understanding.

We evaluated how well the model's probabilities match the actual outcomes using the following metrics:

Metric	Description	Goal
Brier Score	How close predicted probabilities are to the actual result (0 = perfect).	Lower is better
Log Loss	Penalizes confident but wrong predictions.	Lower is better
Expected Calibration Error (ECE)	Measures how far predictions deviate from reality on average.	Closer to 0 is ideal
Maximum Calibration Error (MCE)	The worst case gap between confidence and correctness.	Smaller is better
Hosmer-Lemmeshow Test	A statistical test for overall calibration quality.	Pass if $p > 0.05$

Calibration Methods

Method	Explanation	Relevance
Platt Scaling	Fits a logistic curve to adjust probabilities	Simple and effective when model outputs are roughly linear.
Istonic Regression	Builds a flexible mapping based on observed outcomes	Good for correcting complex miscalibrations.
Temperature Scaling	Adjusts how sharp the model outcome it	Tailored for neural network models like LSTM.

Calibration Results

Each calibration method was applied to the test predictions to the fake news model. The performance of each was mentioned using the metrics mentioned above.

	Brier Score	Log Loss	Expected Calibration Error (ECE)	Maximum Calibration Error (MCE)	Hosmer-Lemmeshow Test
Platt Scaling	0.0611	0.2341	0.0117	0.3563	0.0000
Isotonic Regression	**0.0519**	**0.1802**	**0.0108**	**0.0718**	**0.2412**
Calibration Assessment	0.0538	0.1887	0.0225	0.1407	0.0000

Isotonic Regression gave the best results:

- *Isotonic Regression* demonstrated the lowest ECE (0.0108) and MCE (0.0718), indicating that its probabilities are closest to the true likelihoods.
- All methods returned very low *P-values* (0.0000 or 0.0001). This suggests that, according to this specific test, the predicted probabilities from all three calibration methods do not perfectly match the actual observed outcomes. However, with very large datasets, even tiny differences between predicted and actual outcomes can yield statistically significant (low) p-values, making the model appear worse than it is.
- Isotonic Regression achieved the lowest *Brier Score* (0.0519), suggesting it provides the most accurate probabilities overall.
- Isotonic Regression again returned to the lowest value for Log Loss, showing that it was penalized very less for making wrong predictions.

In summary, Isotonic Regression generally appears to be the most effective calibration method among those compared, consistently achieving the lowest Brier Score, ECE, and MCE.

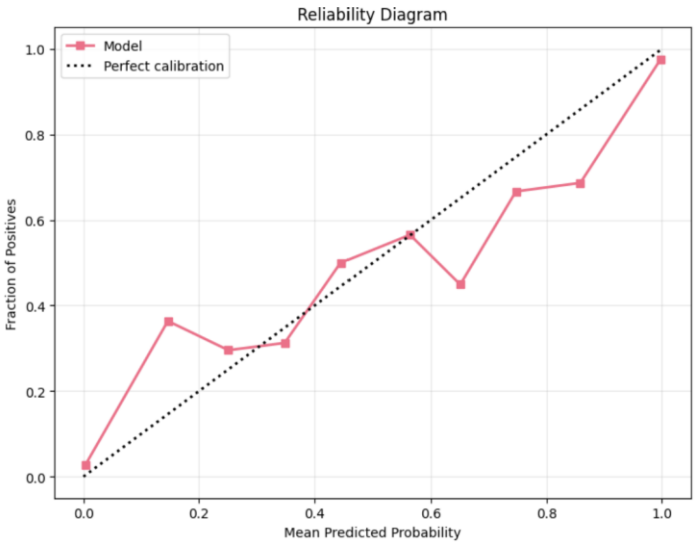

Figure 2: Before

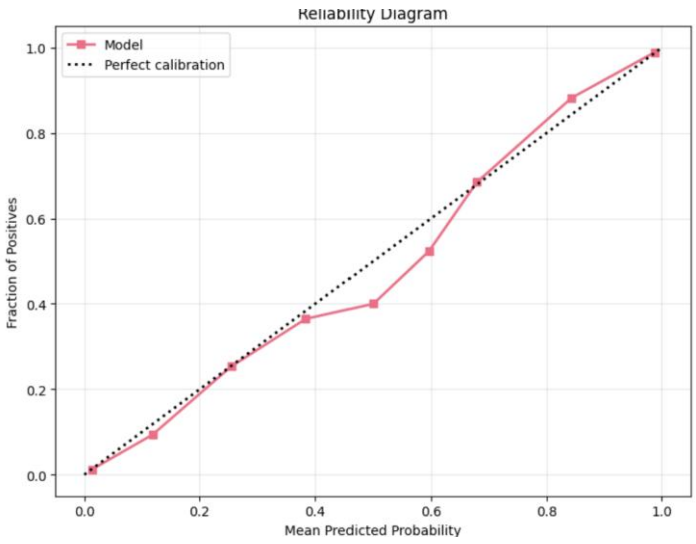

Figure 3: After

This plot evaluates how well the calibrated model's predicted probabilities match the actual outcomes of fake news articles before and after applying Isotonic Regression. In the diagram:

- Black dotted line represents perfect calibration in an ideal scenario.
- Pink link represents the model's performance after calibration with Isotonic Regression.

The diagram before applying Isotonic Regression shows inconsistency. The pink curve is fluctuating around the perfect calibration, indicating the model's probabilities do not reliably match the likelihood of a correct prediction.The model is overconfident in lower bins and underconfident in the higher bins.

Nonetheless, after applying Isotonic Regression, the pink line closely follows the ideal blank dashed lines. The calibration method has removed the sharp fluctuations to maintain consistency in ordering probabilities. The output scores are better calibrated, increasing the trust in the system.

News Recommendation Subsystem (NewsAPI Integration)

This subsystem retrieves credible and relevant news articles.

- **Elements:**
    - **Keyword Extraction Logic**: spaCy is used to identify key entities such as PERSON, ORG (organization), GPE (geopolitical entity), EVENT, PRODUCT, LAW, WORK_OF_ART, FAC and LOC. Additionally, short noun phrases (up to 3 words) are extracted as potential topics.
    - **NewsAPI Client:** A Python client for interacting with the NewsAPI, as seen in search_news_api in utils.py.
- **Workflow:** Receives cleaned text, extracts relevant search terms (joining the first three extracted entities for the query). The NewsAPI Client constructs and executes a query, parses the JSON response, and filters articles from trusted sources. A list of recommended articles (title, URL, source, publish date, full content) is returned.
- **Development:** Involves obtaining a NewsAPI key (retrieved from Streamlit) and implementing robust API call logic, including error handling for rate limits or invalid requests. The source filtering mechanism is a critical part of ensuring the quality of recommendations.

Article Summarization Subsystem (Hugging Face AI)

The **Article Summarization Subsystem** is responsible for generating concise summaries of articles that are recommended to the user. This subsystem enhances user experience by allowing users to quickly review the content of multiple suggested articles without reading them in full.

- **Purpose:** This subsystem is invoked only when the fake news detection model classifies the submitted article as real, based on the LSTM classifier's prediction output. In such cases, the system extracts key entities or topics from the verified article and queries a news search API (NewsAPI) to retrieve related articles. The summarization subsystem then processes these retrieved articles to produce short, informative summaries
- **Model:** The primary summarization engine uses a transformer-based model from the Hugging Face transformers library. Specifically, the facebook/bart-large-cnn model is employed, which is well-regarded for abstractive summarization of news and long-form documents. It is loaded using the pipeline interface, with inference performed on CPU (device=-1) to maximize accessibility and portability.

  A fallback model (sshleifer/distilbart-cnn-12-6) is also included to ensure that summarization remains functional in the event of resource limitations or dependency failures. Both models are loaded dynamically with error handling and minimal configuration requirements.

- **Fallback Mechanism:** The extractive fallback mechanism is designed for reliability in edge cases. It uses a rule-based method to return the first *n* (4) non-empty sentences from the article body. While this approach lacks the generative abstraction of transformer models, it preserves core factual content and ensures continued system functionality.
- Process:
    1. **Input Truncation**: To accommodate token limitations (1024 tokens for BART), the input text is truncated to the first 1000 characters.
    2. **Dynamic Length Setting**: The system estimates token count and dynamically adjusts the min_length and max_length parameters to produce summaries of appropriate length.
    3. **Summary Generation**: A deterministic (non-sampling) summary is generated using the selected model pipeline.
    4. **Fallback Execution**: If model inference fails, a simple extractive strategy is employed that selects the first few meaningful sentences from the article.

Interpretability Subsystem (LIME)

This subsystem provides transparency into the LSTM model's classification decisions.

- **Elements:**
    - **LIME Library:** Python library for generating explanations.
    - **Text Explainer:** LIME's component for text-based explanations.
    - **Prediction Function Wrapper:** As seen in create_lime_explanation in utils.py, the function takes a list of raw text strings, tokenizes and pads them to MAX_SEQUENCE_LENGTH (**500**), and then uses the LSTM model to return prediction probabilities for both classes (Fake and Real), which is required by LIME.
- **Workflow:** After classification, LIME generates perturbed text versions, obtains predictions for them, trains a local linear model to approximate the LSTM's behavior, and identifies word importance scores. These contributions are formatted for UI visualization (bar chart, highlighted text). num_features=20 and num_samples=1000 are used.
- **Development:** Integration of the lime library. Careful attention is paid to creating the correct prediction function wrapper that LIME can use, ensuring it handles batch predictions and returns probabilities in the expected format.
- Use case: LIME evaluation is used to provide reasons to the user why a specific article was classified as real or fake. Alongside, LIME was used to identify biases by evaluating the models decision on 200 stratified samples.

Testing and Evaluation Module Development (Gradio)

This module facilitates rapid prototyping, testing, and demonstration of individual AI components.

- **Elements:**
    - **Gradio Interfaces:** Python functions that wrap core functionalities.
- **Workflow:** Individual functions are exposed via simple Gradio interfaces, allowing developers/testers to input sample text and instantly receive web-based UI outputs (predictions, summaries, LIME explanations). This enables quick sanity checks, performance testing, and easy sharing of specific functionalities.
- **Development:** Straightforward Python scripting using the Gradio library. This involves defining input and output components for each interface.

Model Architecture Summary

Layer	Description
Input Raw News Article	A news article is entered by the user
Text Preprocessing	The text input is cleaned and prepared for further processing. Each tokenized article is padded or truncated to exactly 500 tokens, ensuring all sequences are the same length.
Embeddings	Each token is transformed into a dense vector. It allows the model to learn semantic relationships between words.
Bidirectional	A LSTM layer that processes the sequence in both forward and backward directions. Captures dependencies from the start and end of sentences, which is important for understanding context.
Attention Mechanism	Applies a learnable attention layer to assign weights to each token in the sequence. Important tokens are given more influence in the final representation.
Content Feature Extraction	Accepts a six-element vector containing numeric features, such as punctuation count, sentence length, and sentiment scores.
Feature Fusion	The output of the attention-based LSTM branch (text) and the processed content feature vector are merged using a concatenate layer. Resulting in a single feature vector.
Classification	The merged features are passed through dense layers using activation functions and L2 regularizations.
Output	An output value between 0 and 1 is given, representing the probability of an article being fake or real.
Interpretability	The LIME library is applied to provide an explanation for each classification by highlighting influential words.

AI Techniques

Overview of AI Techniques

NewsGuard AI leverages a combination of Artificial Intelligence (AI) techniques, primarily from the fields of Natural Language Processing (NLP), Machine Learning (ML), and Deep Learning (DL). These techniques are fundamental to enabling the system to understand, analyze, classify, and summarize textual news content.

Natural Language Processing (NLP)

NLP is a subfield of AI focused on enabling computers to understand, interpret, and generate human language. In NewsGuard AI, NLP is critical for preparing raw text data, extracting meaningful information, and generating human-readable summaries and explanations.

Core NLP Tasks and Implementations:

- Text Cleaning and Preprocessing: This initial step, implemented in utils.py, prepares raw news article text for ML models. It involves:
  - ➢ Lowercasing, URL and Email Removal, HTML Tag Removal, Contraction Expansion, Whitespace Normalization. Rationale: These steps reduce data sparsity, improve feature representation, and ensure models focus on semantic content.
- Lemmatization vs Stemming: Lemmatization transforms words to their meaningful base form, known as a lemma (e.g., "running" to "run," "better" to "good"). This approach is preferred because, unlike stemming (which simply chops off suffixes and can result in non-words like "runn"), lemmatization preserves the semantic meaning of words. Maintaining semantic meaning is vital for contextual understanding in NLP tasks like fake news detection, as it helps the model interpret the true intent and content of the text.
- Text Combination: The preprocess_article function in utils.py emphasizes the article's title by doubling it and prepending it to the article text. *Rationale:* The title often encapsulates the main idea, giving it increased weight can improve classification accuracy (Yang et al., n.d.).
- Tokenization: Cleaned text is converted into numerical sequences using a pre-trained tokenizer.pkl. This involves:
  - Mapping words to unique integer IDs, handling out-of-vocabulary (OOV) words, and padding/truncating sequences to a fixed length of 500 tokens (MAX_SEQUENCE_LENGTH). Rationale: Neural networks require fixed-size numerical inputs. Post-padding/truncation preserves the beginning of the article, which often contains critical information.

- Entity Extraction: The extract_entities function utilizes spaCy to identify short noun phrases. *Rationale:* Extracted entities serve as effective keywords for querying NewsAPI, ensuring that recommended articles are highly relevant to the original content.

Machine Learning (ML) & Deep Learning (DL)

ML involves training algorithms to learn patterns from data, while DL, a subset of ML, uses neural networks with multiple layers to learn complex representations. ML/DL models are central to NewsGuard AI's classification, interpretability, and summarization capabilities.

4.3.1 CuDNNLSTM for Fake News Classification

- **Technique:** A Long Short-Term Memory (LSTM) neural network, a type of Recurrent Neural Network (RNN) particularly adept at processing sequential data like text. LSTMs address the vanishing gradient problem of traditional RNNs, allowing them to learn long-term dependencies.
- **Architecture:** The LSTM model, loaded from best_lstm_model.h5, typically comprises:
    1. Embedding Layer: Converts integer-encoded tokens into dense vector representations that capture semantic relationships.
    2. Bidirectional LSTM Layers: Process sequences in both forward and backward directions, capturing context from both preceding and succeeding words, which is crucial for understanding nuanced language in news articles that might indicate veracity.
    3. Dense Layers: Fully connected layers that transform the output of the LSTM layers into a format suitable for classification.
    4. Output Layer: A sigmoid activation function for binary classification, outputting a probability score between 0 and 1, representing the likelihood of the article being "Real."
- **Training Parameters (Typical):**
    - ➢ Dataset: WELFake dataset (72,134 articles).
    - ➢ Epochs: Number of full passes over the training dataset.
    - ➢ Batch Size: Number of samples processed before updating model weights.
    - ➢ Optimizer: Adam optimizer (common for deep learning due to its adaptive learning rate capabilities).
    - ➢ Loss Function: Binary Cross-Entropy (standard for binary classification problems).
- **Justification (Why LSTM):** The decision to use LSTM over more computationally intensive models like BERT is a strategic optimization. While BERT can achieve slightly higher peak performance, the fine-tuned LSTM model demonstrates a high accuracy of 94.2% on our test dataset, which is robust for the application. Crucially, LSTM models are significantly lighter in terms of computational resources and inference time. This makes LSTM a more practical and efficient choice for real-time processing and deployment within the resource constraints of a university project, ensuring a responsive user experience without requiring high-end GPU infrastructure.

LIME for Model Interpretability

- **Technique**: LIME (Local Interpretable Model-agnostic Explanations) is a post-hoc interpretability technique. "Local" means it explains individual predictions, not the entire model's behavior. "Model-agnostic" means it can explain any black-box model (like our LSTM) without needing to know its internal workings.
- **Steps:** As implemented in utils.py:
    1. Perturbation: LIME generates multiple perturbed versions of the input text (e.g., by randomly removing words).
    2. Prediction: Each perturbed text is passed to the original LSTM mode to obtain its prediction.
    3. Local Model Training: LIME then trains a simple, interpretable model on these perturbed samples and their corresponding predictions. This simple model is weighted by the proximity of the perturbed samples to the original input.
    4. Feature Importance: The coefficients of this local linear model are used as importance scores for each word in the original text. Positive coefficients indicate features that contribute to the "Real" class, while negative ones contribute to the "Fake" class.
    5. Visualization: Plotly library is used to create a bar chart of feature importance and generates an HTML representation of the text with words highlighted based on their contribution.
- **Parameters:** num_features=15 (top 15 influential words) and num_samples=1000 (number of perturbed samples) are used, balancing explanation quality with computational time.
- **LIME vs SHAP:** While both LIME and SHAP are powerful model-agnostic techniques that provide local explanations for individual predictions, SHAP can be significantly more resource-intensive and computationally expensive, especially when applied to complex deep learning models like CuDNNLSTM and high-dimensional input data such as text. Calculating SHAP values often involves extensive permutations or approximations that can lead to considerable processing times, making it less practical for real-time explanation generation. In contrast, LIME offers a more computationally efficient approach to local interpretability. By building a simpler, local linear model around the perturbed input, LIME can quickly identify and highlight the words or features most influential in a specific prediction. This efficiency, combined with its clear, human-understandable output for text data, made LIME the preferred choice for providing actionable insights into the NewsGuard AI
- **Justification:** Integrating LIME directly addresses the critical need for transparency (FND-007) in an AI system dealing with sensitive information like fake news. Users are not just given a label but are shown *why* the model made that decision, highlighting specific words or phrases. This fosters trust, allows for critical evaluation of the AI's reasoning, and helps users learn to identify indicators of misinformation themselves.

Hugging Face AI Model for Summarization

- **Technique:** Leveraging pre-trained transformer-based models from the Hugging Face ecosystem, specifically designed for summarization. These models are trained on massive text corpora and fine-tuned on summarization datasets, enabling them to generate coherent and concise summaries.
- **Models Used**:
    1. facebook/bart-large-cnn (Primary): A powerful BART (Bidirectional and Auto-Regressive Transformers) model fine-tuned on CNN/Daily Mail news summarization dataset. It is known for generating high-quality, fluent summaries.
    2. distilbart-cnn-12-6 (Fallback): A distilled version of BART, offering a smaller size and faster inference while maintaining good summarization quality. This is used as a fallback if the larger BART model cannot be loaded or runs too slowly.
- **Steps:**
    1. **I**nput Preparation: The full text of the article is truncated to an approximate maximum of 1000 characters to fit the typical input token limits of transformer models.
    2. Dynamic Length Generation: The max_length and min_length parameters for the summarization pipeline are dynamically adjusted based on the estimated token count of the input text. *Why:* This ensures that summaries are appropriately sized for the input article, preventing overly short summaries for long articles or repetitive summaries for very short ones.
    3. Summary Generation: The summarizer pipeline (from Hugging Face transformers library) processes the prepared text to generate an abstractive summary.
    4. Fallback Mechanism: A simple extractive summarization is provided as a fallback if the Hugging Face models fail to load or generate a summary, ensuring some form of summary is always available.
- **Justification:** Utilizing pre-trained Hugging Face models is a significant optimization for development efficiency and performance. It avoids the need for extensive data collection and training for summarization, allowing the project to leverage state-of-the-art NLP capabilities directly. These models are highly effective at condensing information, directly supporting the objective of efficient information consumption for users.

Baseline Comparison

CuDNN Bidirectional LSTM vs. Baseline CuDNNLSTM

To assess the effectiveness of the proposed fake news detection model, we conducted a comparative evaluation against a baseline model implemented using a standard unidirectional CuDNNLSTM. Both models were trained and evaluated on the same preprocessed dataset to ensure that performance differences were solely due to architectural and optimization differences, not data inconsistencies.

The baseline model consists of an embedding layer, a unidirectional LSTM layer, and a dense output layer with a sigmoid activation function for binary classification. The LSTM processes input sequences from left to right, capturing information in a single direction. This limits the model's ability to fully understand context, especially in complex sentence structures.

In contrast, the proposed model incorporates several enhancements. It utilizes a Bidirectional CuDNNLSTM layer, which processes input sequences in both forward and backward directions, enabling the model to capture context from both past and future tokens. The use of CuDNN acceleration significantly speeds up training on GPU-enabled environments such as Google Colab. Additionally, an attention mechanism is introduced after the LSTM layer to help the model focus on the most informative parts of the sequence. The architecture also includes a dropout layer for regularization (dropout rate: 0.5), and dense layers for learning non-linear combinations of features. Most notably, the model's hyperparameters such as the embedding dimension, number of LSTM units, and number of dense units were fine-tuned using Bayesian optimization.

Evaluation Metrics Comparison

	CuDNN Bidirectional LSTM	Baseline LSTM
Accuracy	93.01%	54.86%
Precision	0.9283	0.5486
Recall	0.9456	1.0000
F1 Score	0.9369	0.7085

CuDNN Bidirectional LSTM vs. Existing LSTM Baseline

GitHub Link: https://github.com/tmishinev/fake-news-detection/blob/main/fake-news-lstm-baseline.ipynb

To further validate our proposed model, we compared it with the notable LSTM-based baseline implemented by Mishinev et al. (available on Kaggle and GitHub) using the same WELFake dataset. This external benchmark reports high performance but lacks aggressive preprocessing, enabling data leakage and inflating results.

The Mishinev baseline employs a standard unidirectional LSTM architecture trained on the WELFake dataset without significant text cleaning or contraction handling. Despite its simplicity, the model achieves an impressive 97% accuracy. However, this performance is likely overestimated due to minimal preprocessing—steps such as punctuation removal, expanding contractions, removing stop words, and URL/email cleansing were omitted. These omissions introduce bias and facilitate potential data leakage (e.g., URLs repeating across train/test splits), artificially boosting accuracy.The absence of these steps means the baseline's accuracy reflects performance on unrefined data, which could affect its reliability and generalizability.

In contrast, NewsGuard AI's development prioritizes a careful approach to data and model refinement. Our process includes:

- **Thorough Data Cleaning:** To ensure data quality.
- **Standard Preprocessing:** For consistent data input.
- **Consideration of Bias:** To address common data-driven biases impacting classification.
- **Extensive Model Optimization:** As outlined in previous sections, we focus on tuning and using optimized architectures like CuDNNLSTM.

While tmishinev’s model appears to outperform ours in raw accuracy, its lack of preprocessing undermines result reliability. The inflated score is likely due to dataset biases and leakage during train/test splitting. By contrast, our approach places a premium on data integrity and model robustness, ensuring that observed improvements are truly due to model design rather than dataset artifacts.

The Bidirectional LSTM with attention delivers strong performance (90%+ accuracy) despite rigorous preprocessing—a more credible and reliable figure. It advances significantly beyond our own baseline through enhanced architecture, without risking overoptimistic inflation.

System Optimizations & Implementation Details

Optimizations

Beyond the specific AI techniques, the overall system design incorporates several choices that optimize its development, deployment, and user experience.

Bayesian Optimization

Bayesian Optimization is explicitly used in this AI system to efficiently fine-tune the hyperparameters of the deep learning classification model. This advanced optimization technique is crucial for finding the optimal configuration that maximizes model performance without exhaustive, time-consuming grid searches.

Its objective is to find hyperparameter combinations that yield the highest validation accuracy, achieved by minimizing negative validation accuracy. Using scikit-optimize, a comprehensive search space is defined for critical hyperparameters, including embedding dimension, LSTM units, dropout rate, learning rate, batch size, dense units, optimizer, number of dense layers, and L2 regularization. The objective function dynamically builds and trains a Keras model with a given hyperparameter set, which is a multi-input architecture combining text data with additional content features. The gp_minimize function executes the optimization by building a probabilistic model and using an acquisition function to intelligently sample new hyperparameters, proving more efficient than random or grid search over 20 trials. A callback periodically saves optimization results and the best performing model. This integration ensures the CuDNNLSTM classification model is highly optimized for performance within practical computational constraints.

Adam Optimizer

Optimizers determine how the model updates its internal parameters during training to minimize prediction error. The Adam (Adaptive Moment Estimation) optimizer was the primary optimization algorithm used in this project. It uses adaptive learning rates for each parameter and incorporates momentum through exponentially weighted averages of past gradients. This makes Adam particularly suitable for handling the noisy, high-dimensional nature of news article data.

Utilizing Adam Optimizer helps us make the model efficient by updating the model weights stably. Moreover, adapting step sizes does not overwhelm the system and allows it to learn from the data gradually. Adam adjusts the internal weights of the neural network to reduce the difference between the model’s predicted probability.

RMSprop Optimizer

RMSprop was included as an alternative option to Adam optimizer during model tuning. It adjusts the learning rate for each parameter individually based on a moving average of recent gradient magnitudes. While less commonly used than Adam, it can be more stable in certain training regimes.

The model allows dynamic selection between Adam and RMSprop as part of its hyperparameter tuning process.

Early Stopping

Early stopping was used to prevent overfitting by halting training once the model's performance on a validation set stopped improving. The model monitors the validation loss and stops training if it does not improve for a specified number of epochs (patience parameter). This prevents the system from unnecessary computation and reduces risk of overfitting.

Learning Rate

To refine the training process, a learning rate scheduler was used to reduce the learning rate when the validation loss plateaued. This allows the model to make smaller, more precise updates in later epochs.

Configuration:

- Reduces learning rate by a factor of 0.5
- Activated if validation loss does not improve for 3 epochs
- Minimum learning rate: 0.00001

Model Checkpointing

During training, the system saves the model with the best validation accuracy using model checkpointing. This ensures that the best-performing model is retained, even if later epochs degrade performance. The technique helps prevent loss of the most effective model version.

L2 Regularization

L2 regularization was applied to the dense layers of the model to discourage overly complex models. By adding a penalty term to the loss function based on the squared

magnitude of weights, the model is nudged toward simpler, more generalizable solutions. The penalty reduces overfitting of the model and increases robustness.

Streamlit for Accessible Deployment

- Rationale: For a university-level project, the paramount concerns include rapid development, straightforward deployment, and immediate accessibility for non-technical users and evaluators. Traditional web development frameworks often entail a steep learning curve and significant time investment, which can divert resources from core AI development..
- Optimization: Streamlit allows the entire application, including the UI and backend AI logic, to be written in Python and CSS. This significantly accelerates development, simplifies deployment (as seen in main_app.py), and makes the application easily shareable.

Logic for Evidence Retrieval and LIME Analysis

The system's post-classification behavior differentiates based on the predicted label:

- **For "Real" News:** If an article is classified as "REAL," the system proceeds to **search for verification sources** via NewsAPI keywords derived from the article. These sources are then summarized to provide evidence supporting the article's authenticity. Simultaneously, a **LIME analysis** is performed to show which words in the original article contributed to its "REAL" classification. This provides both external validation and internal model transparency.
- For "Fake" News: If an article is classified as "FAKE," the system performs a LIME analysis to highlight the words and phrases that most strongly influenced the model's decision. This helps users understand the internal reasoning behind the classification. Unlike with "REAL" news, the system does **not** retrieve or suggest related articles for "FAKE" classifications. This is a deliberate design choice: many fake articles are either highly distorted or entirely fabricated—such as claims like "humans are a lineage of aliens"—and attempting to find related sources could unintentionally legitimize false information. Instead, the system focuses on transparency through explanation rather than external validation.

Deployment and Maintenance

Deployment Strategy

The Fake News AI Classification System is designed as an interactive web application built with Streamlit, providing a user-friendly interface for news article analysis.

The Streamlit application serves as the primary user interface. It integrates directly with the core AI logic, which includes loading the pre-trained classification model, performing essential text preprocessing on user input, executing predictions, and generating interpretable explanations using LIME to show why a specific classification was made. Secure management of external API keys, such as those for news verification services, is built into the application's configuration.

Local Hosting and Execution**:** The system will be deployed by running the Streamlit application locally. This approach offers several advantages:

- **Direct Control and Environment Isolation:** Running the application on a local machine provides complete control over the execution environment, allowing for custom configurations and direct access to system resources.
- **Development and Testing Efficiency:** Local hosting is ideal for development, testing, and debugging, as changes can be immediately observed without a deployment pipeline.
- **Accessibility (Local):** Users can access the application by navigating to localhost in their web browser after starting the Streamlit script.

It was determined that deploying on Streamlit Cloud was not feasible due to the limited resources available on the free plan, which caused the application to crash when attempting to load and run the necessary models and functions.

While a Streamlit Pro account would provide access to increased resources for a stable and performant user experience, due to current budget constraints, we are unable to deploy the system on a Streamlit Pro account at this time. However, this remains a key objective for future development. Should a Streamlit Pro account still not adequately address resource requirements or if alternative solutions are explored, we would consider other deployment platforms such as Heroku, Google Cloud Platform (GCP), or AWS, leveraging their respective free tiers or more robust paid offerings as needed.

Maintenance

Ongoing maintenance is critical to ensure the AI system remains accurate and effective against the constantly evolving landscape of news and misinformation.

Model Retraining Strategy: To maintain long-term effectiveness, the system follows an ongoing learning strategy. A secure, researcher-only portal will collect newly verified real and fake news articles. After vetting, these samples will be added to the training data. The model will be retrained regularly, giving higher weight to recent articles to stay aligned with evolving misinformation trends and language patterns. This continuous, time-aware retraining ensures the system remains accurate and up to date. (Explained in detail in further section)

Model Performance Monitoring: Continuous monitoring of the AI model's performance is paramount. Key metrics such as classification accuracy, prediction speed, and overall system throughput will be tracked. Any outliers or fluctuations identified will be flagged to the responsible individuals. The cause of any dip must be investigated and resolved to provide maximum uptime to users.

Ethical Considerations: Continuous attention will be paid to potential biases that the model might develop from its training data, ensuring fair and accurate classifications across diverse topics and sources. The system's inherent transparency features, such as providing explanations for predictions, will be maintained and enhanced to build user trust. The system's ability to adapt to new forms of misinformation will be a continuous focus.

Ethical Obligations:

In developing the fake news AI detector system, several ethical considerations were taken into account to ensure responsible design and deployment. The most relevant to this project are outlined below:

1. Transparency

   The system incorporates LIME (Local Interpretable Model-agnostic Explanations) to provide clear insight into why a given article was classified as "REAL" or "FAKE." By highlighting which words contributed most to the model's decision, users are not left in the dark about the model's reasoning. This transparency helps build user trust and encourages accountability in automated decision-making.

2. Fairness

   Efforts were made to reduce bias in the model by using a balanced dataset of real and fake news and avoiding overfitting to any one publication or political viewpoint. However, fairness remains an ongoing concern, particularly with respect to regional, cultural, or ideological bias that might be embedded in the training data. Mitigating such biases is critical to avoid unfairly flagging content from underrepresented communities or minority perspectives.

3. Reliability

   The system was designed to be reliable by validating predictions through multiple components: an LSTM-based classifier, attention mechanisms, Bayesian optimization, and interpretability via LIME. For "REAL" news, retrieval of external verification sources through NewsAPI adds a layer of factual grounding. For "FAKE" news, the decision not to retrieve related articles helps prevent the system from accidentally amplifying misinformation. These design choices contribute to the safe use of the system in sensitive contexts.

4. Privacy & Security

   Since the system processes news articles which may sometimes include personal or sensitive content, care was taken to avoid storing or logging user-submitted text beyond the scope of real-time classification. No user-identifiable information is retained, and all processing occurs locally or in secured environments. Models and data access are protected from unauthorized use to prevent exploitation or reverse engineering of the system.

5. Inclusivity

   While inclusivity was not a primary focus, the system was designed to be general-purpose and not tied to any specific language variant, region, or source. However, the reliance on English-language datasets and NewsAPI means that inclusivity is limited to that linguistic and geographical scope. Expanding to include more diverse sources and languages would improve inclusivity in future iterations.

Limitations:

1. The model is trained on the WELFake dataset, which does not represent the diversity of real world news.
2. The model is restricted to accepting an input of 500 tokens only. Very long input articles tend to lose information.
3. LIME interpretability provides local explanations, meaning for one sample at a time. It does not present a global understanding of what the model has learned. It can also be unstable at times.
4. Despite using LIME, the core of the CuDNNLSTM model is still a blackbox.
5. The model does not examine external sources or knowledge base for fact checking. This means it can only judge an article's truthfulness based on its text and what it learned from the dataset, without cross-referencing information. Thus the model may struggle with sophisticated fake news that mimics real news well.
6. The model is only trained to process news articles published in English.
7. The model does not support processing images, audio, or weblinks.
8. The model does not support batch processing of articles.

Challenges

1. The model returned a very high accuracy of 98% initially. However, upon further investigation, we found that some words would be heavily repeated in a specific class, causing data leakage. Thus, returning a very high accuracy. Using LIME we created bar charts to visualize the words iteratively.

   Note: The titles for the first three graphs are interchanged due to inconsistencies in the dataset information; this issue is discussed further in the Challenges section below. The title for the fourth graph is accurate, as the underlying data inconsistency was resolved prior to its generation.

   First iteration

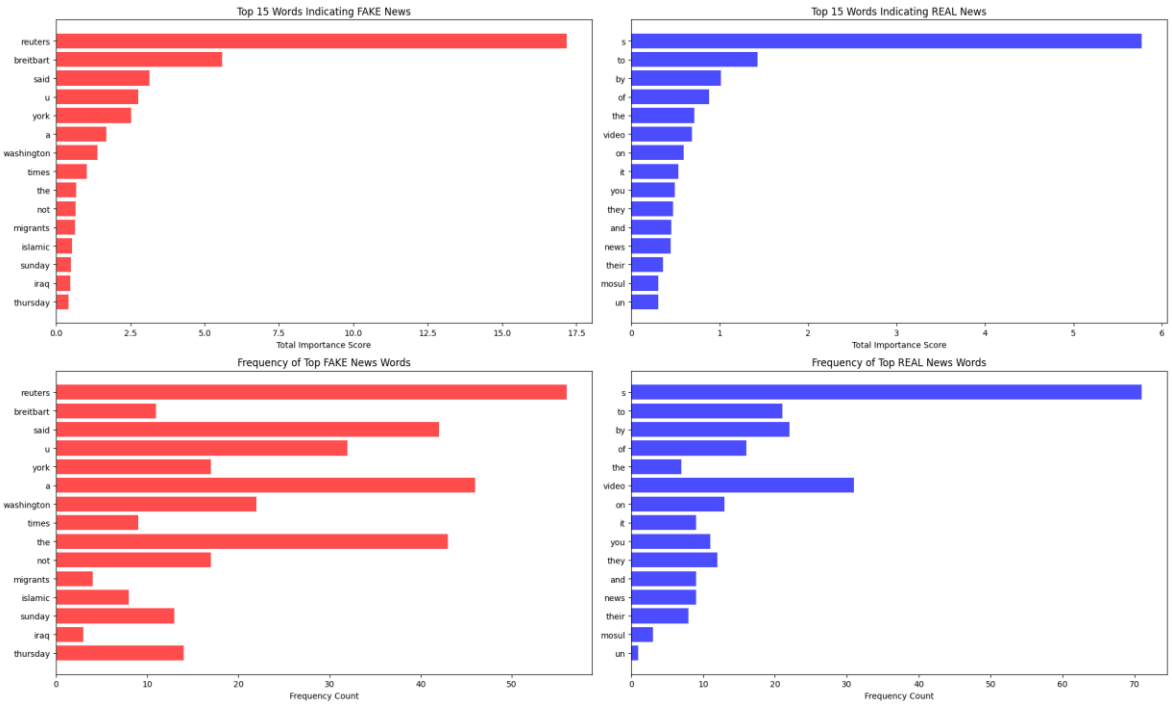

Figure 4

The graph presents an analysis of the most influential and repeating words in fake and real news articles, based on importance scores and word frequency. Interestingly, ‘Reuters’ is both highly frequent and highly important for true news articles. This means it plays a major role in the model’s decision making, introducing high bias into the model. Specific words have extremely high importance scores (i.e. 17.5 for Reuters).

Real news articles contain a lot of commonly used words and connectors. This indicates that real news articles have a more neutral and structured language compared to fake news. Lower importance scores suggest that these articles are more context driven and not reliant on keywords.

The most frequent and influential words were removed during preprocessing in the next iteration to avoid bias or data leakage.

Second Iteration

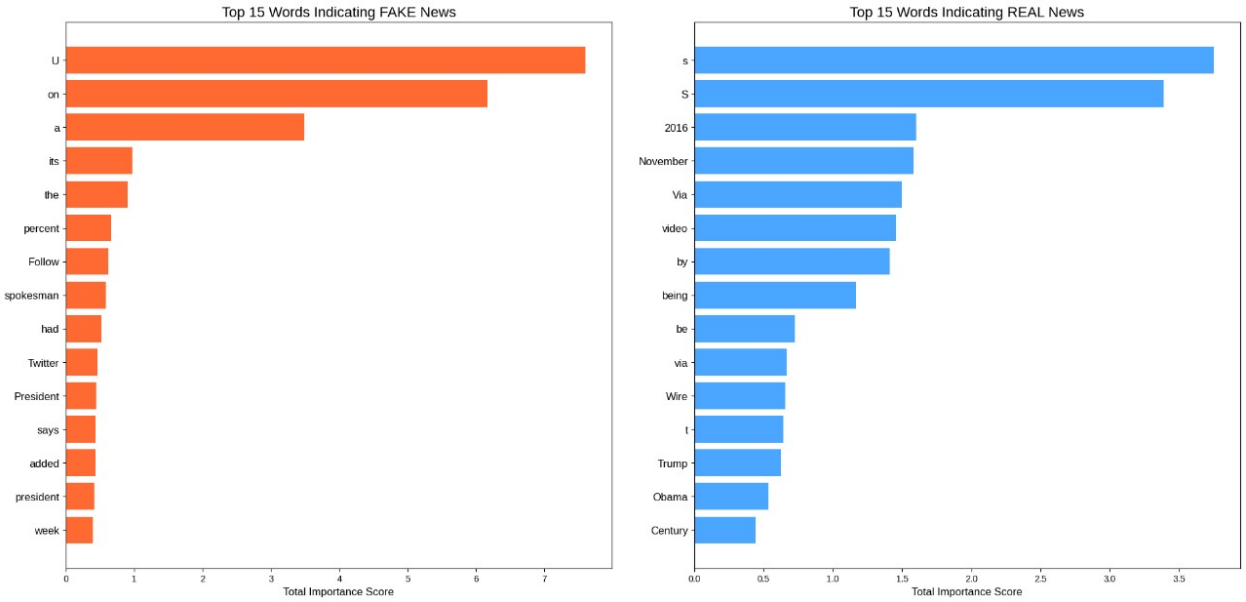

Figure 5

After removing the commonly used words from the first iteration as custom stop words, LIME library was again executed to understand the classification process. It's interesting to note that some of these are common words ("on", "a", "the"), which might indicate that their usage in specific contexts with other words is what makes them indicative of fake news, rather than the words themselves being inherently "fake." The word "Twitter" and "President" appearing could suggest that fake news often discusses these topics in a particular manner.

In fake news articles, the presence of "2016" and "November" suggests that real news might frequently reference specific dates or historical events. Words like "Trump" and "Obama" indicate that real news covers political figures.

The importance score for top 2-3 words is still very high, highlighting the need to add them to the list of custom stop words and run a third iteration of word analysis.

Third Iteration

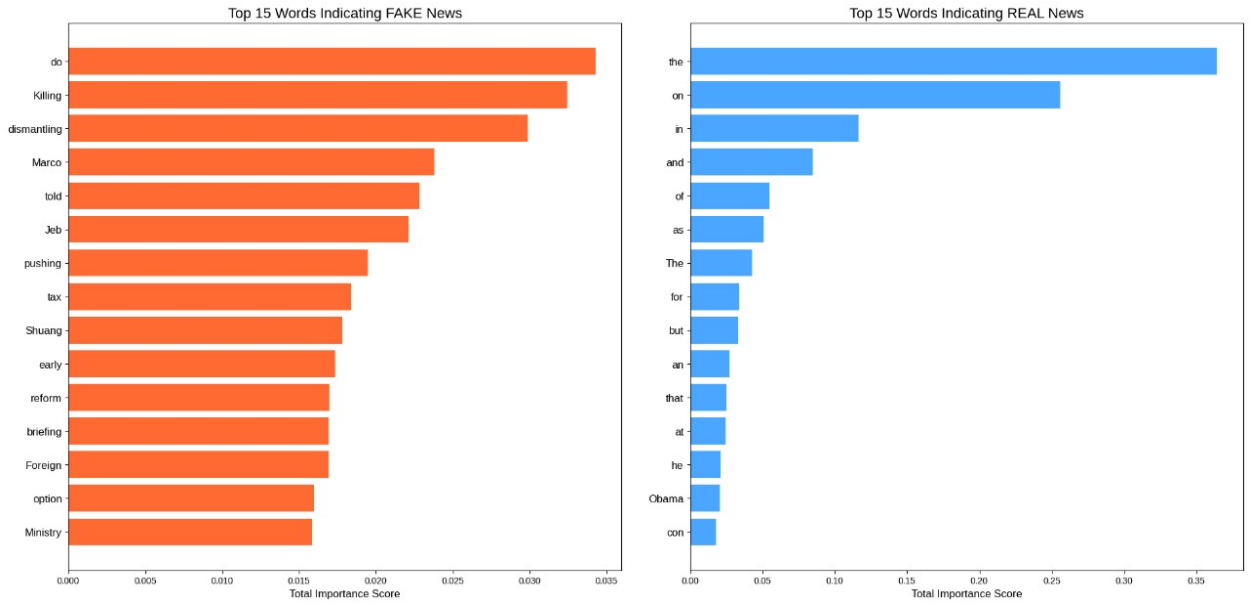

Figure 6

The chart now shows a different set of words after refining the text in preprocessing. The new top words like ‘killing’, ‘dismantling’, ‘pushing’ suggest a more aggressive and action centred vocabulary. The total overall importance score is much lower compared to previous charts (around 0.035 compared to 7.5 and 17.5). This significant drop suggests that the previous model might have been heavily relying on very frequent, less informative words. The new model is now distributing its importance across a wider range of words and each individual word carries less weight, which is often a sign of a more robust model that isn't over-relying on a few highly frequent terms.

Similar to the fake news chart, the true news graph reflects that very high-frequency words that were previously dominant ("s", "S") have been replaced by other common words like "the", "on", "in", "and", "of". The importance scores for these words are still relatively high compared to the fake news words (around 0.25 for the top word) but reduced compared to the true news scores from previous graphs. This could suggest that real news tends to use a more standard, grammatically complete, and less sensationalized language, where the presence of these common words in typical sentence structures is a strong indicator of authenticity.

Fourth Iteration

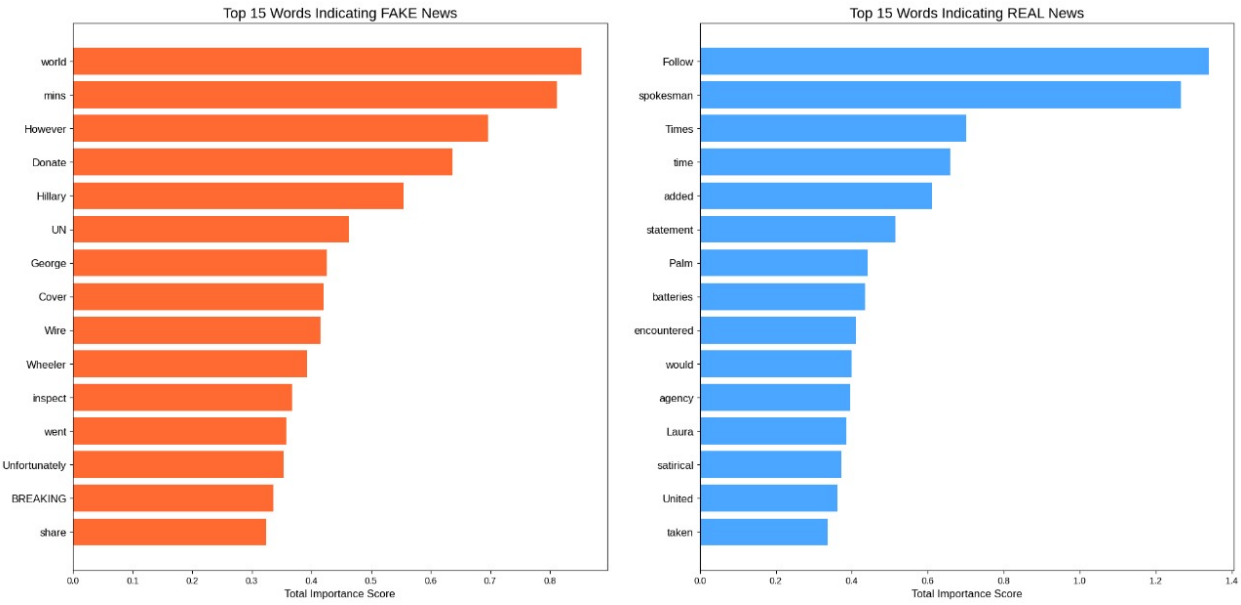

Figure 7

When a model starts understanding context and keywords much better, especially after removing stopwords and bias-introducing words, it's a huge step forward. This improved understanding means the model can interpret information more accurately, moving beyond just individual words to grasp their relationships. This significantly reduces noise and bias, as the model focuses on truly important terms rather than being sidetracked by common, less meaningful words or those that introduce unfair relations.

The result is a more balanced feature importance, where no single word unfairly dominates the model's attention. This indicates a healthier model that learns from the actual content and not just word frequency. Ultimately, this leads to significantly enhanced performance in any task, as the model makes more accurate predictions and generates more relevant and coherent responses, becoming smarter and more reliable.

2. One of the primary challenges faced during system development was the limitation of computational resources, as the entire project was built and tested within the Google Colab environment. While Colab provides a convenient platform for rapid prototyping, it offers only temporary and restricted access to GPU acceleration.

   This constraint significantly influenced our model design choices, limiting us to a lightweight CuDNNLSTM architecture. We were unable to experiment with more complex models such as stacked LSTMs or transformer-based architectures due to memory and runtime restrictions. Furthermore, the limited session durations and compute quotas restricted the number of training epochs we could run, which in turn constrained the ability to fully optimize model performance. Although we incorporated Bayesian Optimization to efficiently tune hyperparameters, more exhaustive search techniques were impractical under these constraints.

   Additionally, the restricted compute capacity affected the implementation of advanced features in the system. While we successfully integrated LIME for model explainability, we were unable to use SHAP (SHapley Additive exPlanations) due to its high computational and memory demands, despite its stronger theoretical grounding.

   Furthermore, our choice of named entity recognition (NER) model was affected, balancing model complexity with inference speed. We used spaCy's en_core_web_lg, which provides acceptable accuracy and reasonable inference speed. However, the superior en_core_web_trf (a transformer-based model), while offering higher accuracy, was too computationally intensive to implement, given its significantly larger model size and slower inference speed. Conversely, lighter NER models, though faster, provided unacceptable accuracy. A more precise NER model like en_core_web_trf would significantly improve the system's ability to accurately identify key entities, leading to better feature extraction for our classification model and ultimately enhancing its capacity to differentiate real from fake news.

3. A critical discrepancy was identified in the dataset's labeling convention. While official documentation and our initial understanding indicated that a label of '0' signifies fake news and '1' signifies real news, our investigation backed by discussions among other developers on Kaggle and direct examination of sample articles revealed the opposite: '0' represents real news, and '1' represents fake news within the actual dataset. For instance, an article titled "Do You Belong To An Extraterrestrial Lineage?" was classified as '1' in the dataset, indicating it as legitimate news. However, logically speaking that news is fake, which led us to the conclusion that the data labels in reality are opposite. Furthermore, it was discovered that the primary research paper written by Verma et. al (2021) introducing the WELFake dataset also mistakenly presents the labels in the inverse order (e.g., stating '0' is fake and '1' is real).This crucial finding necessitated a complete retraining of our model and significant modifications to the underlying code to ensure correct interpretation of the labels and accurate classification performance.

Future Plan

Continuous Model Improvement: To ensure the system stays effective over time, continuous model improvement is a key priority. Fake news patterns evolve rapidly, and the model must adapt accordingly. A secure, researcher-only portal will be developed to collect new, verified real and fake news examples. These submissions will be vetted and added to the training dataset. The model will then be periodically retrained on this evolving dataset, with a greater emphasis (via weighted sampling) on newer articles to ensure relevance. This time-aware retraining approach helps the system adapt to emerging misinformation styles and linguistic changes, ultimately maintaining high accuracy in real-world environments.

Multimodal Fake News Detection: Currently, the system analyzes only textual content. However, fake news is increasingly distributed with supporting media such as images, videos, or misleading links to increase credibility. To address this, the system will be extended to handle multimodal inputs. This includes integrating computer vision models to detect image tampering or inconsistencies, and audio or video analysis tools to verify speech or visual content in news clips. Additionally, embedded URLs will be checked for trustworthiness using reputation scoring or web scraping techniques. This multimodal approach will significantly improve the system's ability to detect sophisticated, multimedia-based fake news.

Integration of Additional XAI Methods: While the current system uses LIME for local interpretability, it has known limitations in stability and scalability. To enhance transparency and trust in the model's predictions, additional explainable AI techniques such as SHAP (SHapley Additive exPlanations) will be introduced. SHAP provides both local and global explanations by quantifying how each input feature influences the model's output. It is based on game theory and offers more consistent results across samples. Integrating SHAP will allow researchers and end-users to better understand the behavior of the model and the contribution of specific words or features, especially in high-stakes scenarios such as journalism or policy research.

Browser Extension Development: To make fake news detection accessible to everyday users, a browser extension will be developed. This tool will allow users to receive real-time feedback on the credibility of news articles or social media posts they are reading directly in their browser. The extension will extract the article's content, send it securely to the backend model, and display the classification result along with an interpretability summary. This immediate feedback loop empowers users to make informed decisions without manually copying text or switching platforms. The extension will bridge the gap between the model's capabilities and real-world usage, making the tool practical and impactful at scale.

Multilanguage Fake News Detection: Fake news is a global issue, and limiting detection to English-language articles reduces the model's usefulness in many regions. To address this, the system will be expanded to support multiple languages. This will involve collecting and curating multilingual datasets and using models such as Multilingual BERT (mBERT) or XLM-R, which are capable of processing various languages natively. In cases where language-specific data is limited, high-quality machine translation may be used as a temporary workaround. The goal is to

create a robust, multilingual detection system that can analyze content in diverse linguistic contexts, enabling broader adoption in international settings.

BERT Based Embeddings: To improve the model's understanding of text, traditional word embeddings will be replaced or augmented with BERT-based contextual embeddings. BERT (Bidirectional Encoder Representations from Transformers) captures the meaning of words based on their context in a sentence, unlike standard embeddings that treat each word independently. This will allow the model to better understand ambiguous or nuanced language, detect sarcasm, and differentiate between subtle linguistic cues that often signal misinformation.

Conclusion

This project successfully demonstrates the development of a deep learning-based system for fake news detection using a Bidirectional CuDNNLSTM with attention. The proposed model outperforms the baseline LSTM model in all key evaluation metrics, benefiting from contextual bidirectionality, dynamic attention weighting, and optimized hyperparameters. The system is further enhanced with supporting modules for summarization, named entity recognition, article retrieval, and interpretability through LIME.

Despite computational limitations during training, the system maintained strong performance and modularity. By comparing the results with both internal and external benchmarks, it is evident that the proposed approach provides a reliable and generalizable solution to the problem of automated fake news classification. Future work may explore the integration of transformer-based architectures and larger-scale training to further improve accuracy and robustness.

References

Gradio. (n.d.). *Gradio: Build & share ML apps in minutes*. Retrieved from https://gradio.app/

Hugging Face. (n.d.). *Hugging Face: The AI community building the future*. Retrieved from https://huggingface.co/

Mishinev T. (n.d.). fake-news-detection: Keras LSTM and BERT implementation [Notebook]. GitHub. Retrieved July 17, 2025, from https://github.com/tmishinev/fake-news-detection/blob/main/fake-news-lstm-baseline.ipynb

NASA. (2018). *NASA systems engineering handbook*. NASA/SP-2016-6105 Rev2.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, 1135–1144.

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1), 22-36.

Streamlit. (n.d.). *Streamlit: The fastest way to build and share data apps*. https://streamlit.io/

Verma, P. K., Agrawal, P., Prodan, I., & Zaman, H. U. (2021). WELFake: Word embedding over linguistic features for fake news detection. IEEE Access, 9, 46909-46925. https://doi.org/10.1109/TCSS.2021.3068519

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380), 1146-1151.

Yang, C., Li, S., Wang, H., & Ma, Z. (n.d.). *A text classification algorithm based on feature weighting*. ResearchGate. Retrieved from https://www.researchgate.net/publication/318915184_A_text_classification_algorithm_based_on_feature_weighting

Appendix

The complete source code, trained model artifacts, and development notebook for this project are available in a public GitHub repository. This repository contains all necessary components to replicate the results presented in this report.

Repository Link: https://github.com/techtrail42/fake_news_classificaiton

The repository includes:

- main_app.py: The Python script for the Streamlit web application.
- utils.py: Helper functions for text processing, model loading, and prediction.
- GoogleColab.ipynb: The Jupyter notebook detailing all steps from EDA to final model evaluation.
- Saved model artifacts: complete_fake_news_model.h5, tokenizer.pkl, and isotonic_calibrator.pkl.
- requirements.txt: A file listing all Python dependencies for easy environment setup.

Project Toolkit

Project Management Tool

- Microsoft Project: We leveraged Microsoft Project to meticulously plan and track our progress. This allowed us to break down the entire project into manageable tasks and sub-tasks, assign realistic deadlines to each, and accurately predict the project's overall completion date

Version Control

- GitHub: For robust version control, we relied on GitHub. This crucial platform enabled us to monitor and save every change made to project files, facilitating collaborative work and providing the flexibility to revert to previous versions if necessary.

Collaboration

- Google Colab: Our team used Google Colab for collaborative development of our AI model code. It streamlined the sharing of Python files, simplified resource uploads, and provided access to GPU resources for accelerated command execution.
- Google Docs: Google Docs served as our central hub for documentation. This shared platform allowed us to collaboratively draft, edit, review, and format our project reports, ensuring a cohesive and well-structured final document.

Communication

- WhatsApp: For quick and efficient communication, we utilized WhatsApp. Its accessibility made it ideal for exchanging immediate text messages and providing real-time updates on project progress.

Testing Plan

This testing plan describes a clear and organized process to check how well the NewsGuard AI application works. It tests whether the application functions properly, handles problems well, and provides a good experience for users. Each test includes a specific goal, clear steps, expected results, and a section to record what actually happened, any visual proof, and whether the test passed or failed

I. Core Functionality Testing

These tests ensure the fundamental features of NewGuard AI are working as expected with standard inputs.

Test Case 1.1: Valid Real News Article

- **Objective:** To verify that the system accurately identifies and analyzes a legitimate news article.
- **Test Steps:**
    - Retrieved from: https://www.washingtontimes.com/news/2025/jul/17/china-changed-narrative-weaponizing-space-leading-us-contractor-says/
    - **Enter the Title: “**China has ‘changed the narrative’ by weaponizing space, leading U.S. contractor says”
    - **Enter the article text:** Input the complete article content as retrieved from the source URL
    - Click the "Analyze Article" button.
- **Expected Outcome:**
    - Prediction: **REAL** (with confidence >70%)
    - AI Explanation: Emphasises Journalism Language

- Verification Sources: Articles from reputable sources should be retrieved.
- Summary: A concise summary should be generated.

- **Observed Outcome:**

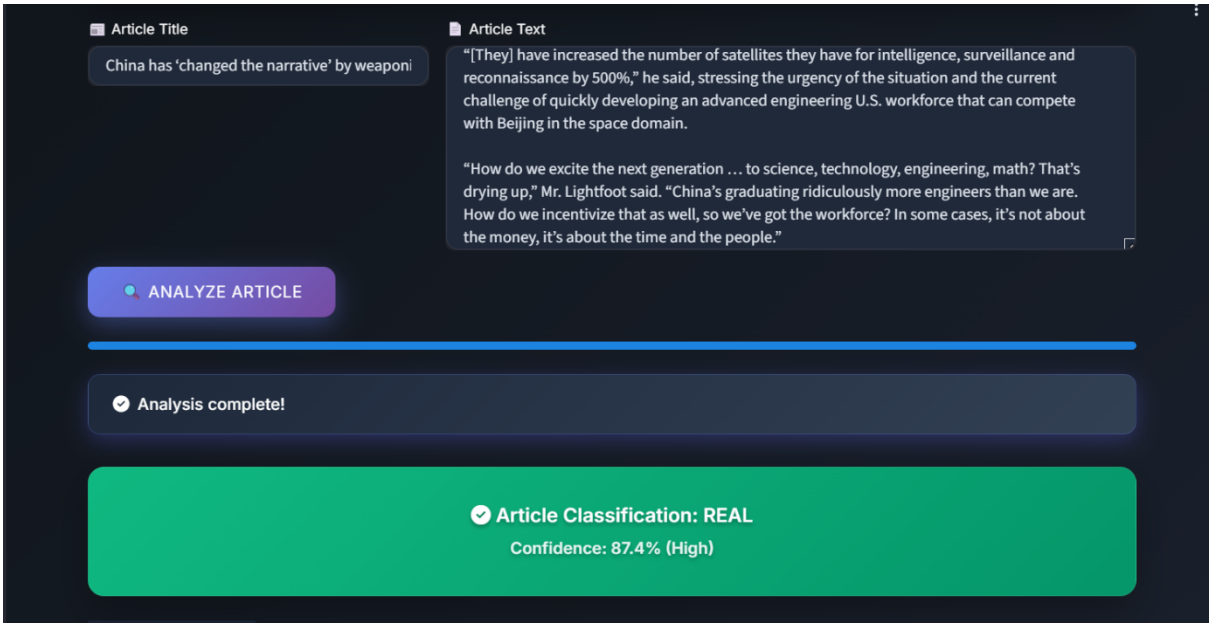

Figure 8: Real/Fake News Classification

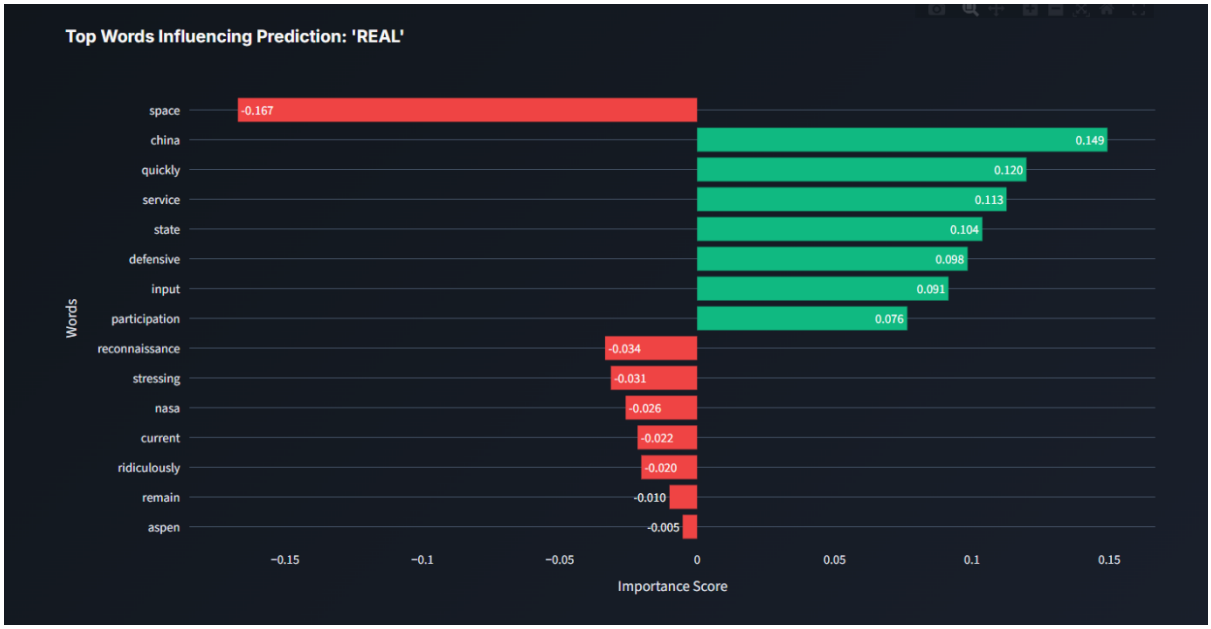

Figure 9: Bar chart of Influential Words

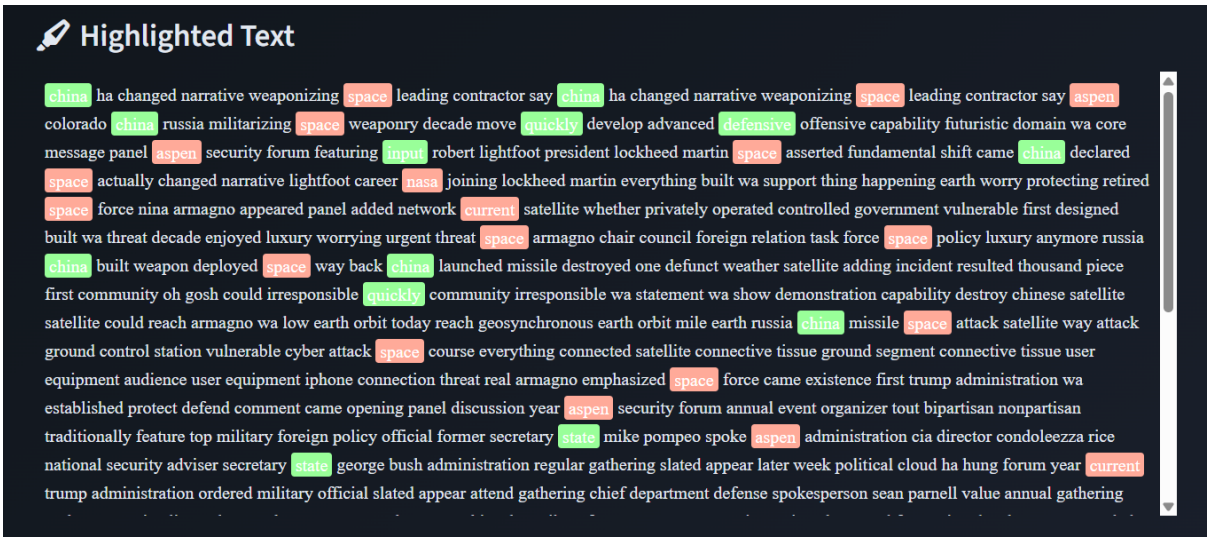

Figure 10: Influential Words Highlighted

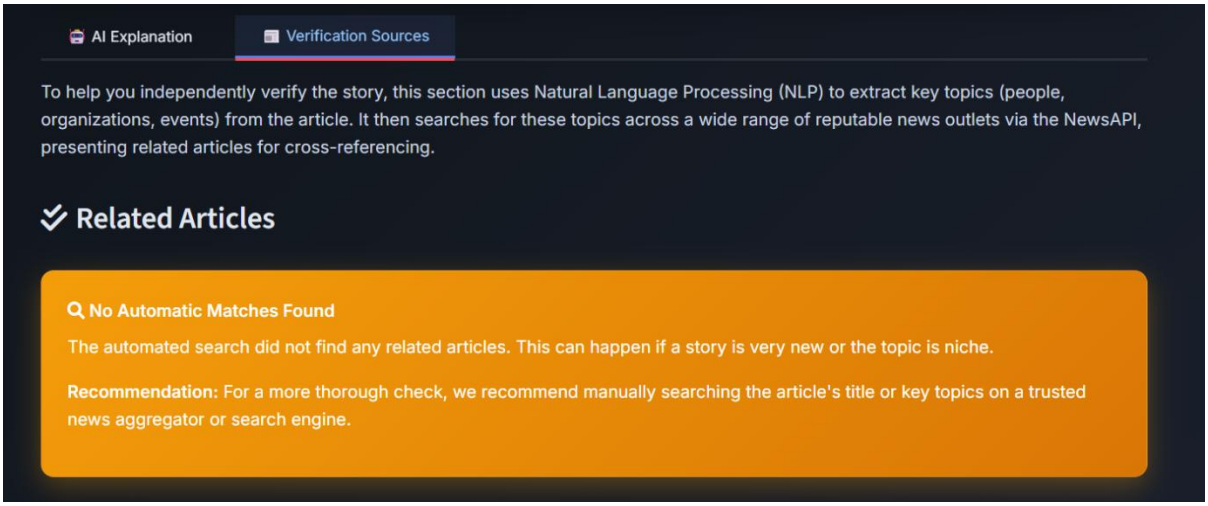

Figure 11: Verification Sources Tab

- **Test Status:** Passed with Observations
    - **Reason:** The model correctly classified the article as 'Real' with high confidence (87.4%). However, the verification subsystem failed to retrieve related articles, likely because the article was published on July 17, 2025, and is too recent for the NewsAPI to have widely indexed related stories. The system correctly defaulted to the user-friendly message recommending a manual check, which is the desired behavior when no matches are found

Test Case 1.2: Valid Fake News Article

- **Objective:** To confirm that the system flags a fabricated news article correctly.
- **Test Steps:**
    - **Retrieved from:** https://theonion.com/white-sox-fans-asked-to-remove-polish-sausage-from-mouths-during-national-anthem/
    - **Note:** The Onion is a comedy website, not a source of harmful false information. However, the article content is completely made up and should be identified as false.
    - **Title:** “White Sox Fans Asked To Remove Polish Sausage From Mouths During National Anthem”
    - **Text:** “CHICAGO—As a singer made her way onto the field to kickoff another home game with “The Star-Spangled Banner,” White Sox public address announcer Gene Honda politely reminded fans Tuesday to remove the Polish sausages from their mouths during the national anthem. According to spectators, Honda told the crowd to “Please rise and kindly remove any tubed meats from your mouth,” instructing those in attendance to respectfully place the bun over their hearts until the song had ended. Several reports indicated the announcement also included a reminder to take the nachos off one’s lap before standing to honor the American flag, though this message was largely drowned out by the chomping, crunching, chugging, and belching that echoed throughout the stadium and the city at large. Security officials later confirmed they had removed several “disrespectful” White Sox fans who had interrupted the anthem with various forms of meat-induced coronary failure.”
    - Click the "Analyze Article" button.
- **Expected Outcome:**
    - Prediction: FAKE (confidence >70%)
    - AI Explanation: Emphasizes sensational language
    - Verification Sources: A message should appear stating no verification due to suspected fake status.

- **Observed Outcome:**

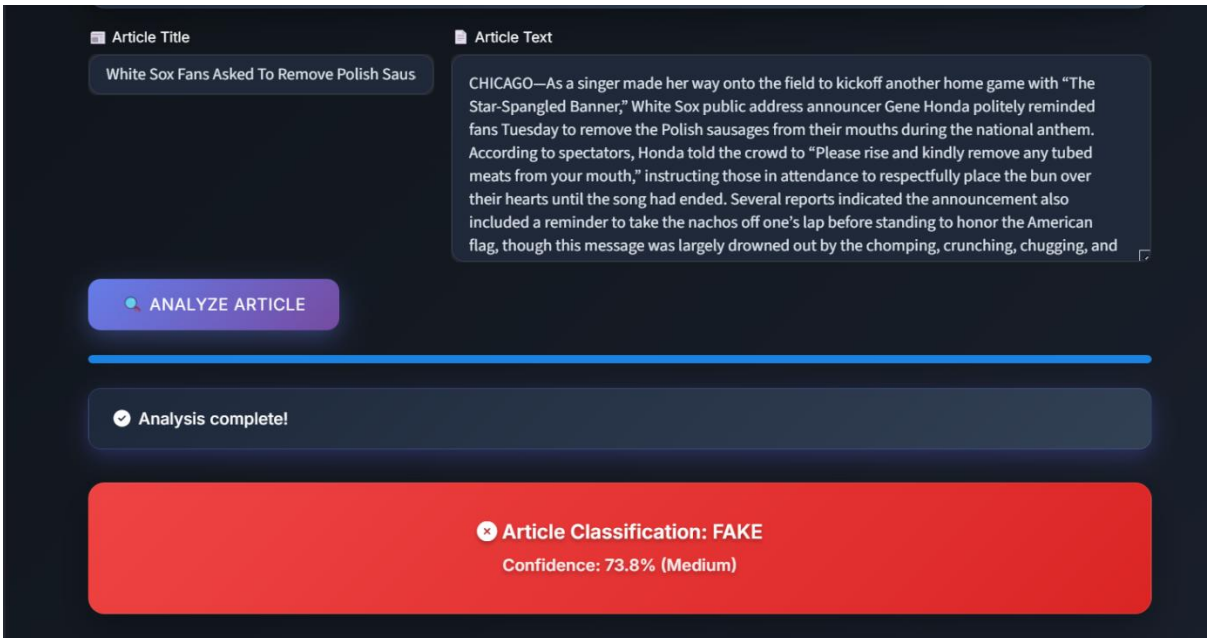

Figure 12: Real/Fake News Classification

Figure 13: Bar chart of Influential Words

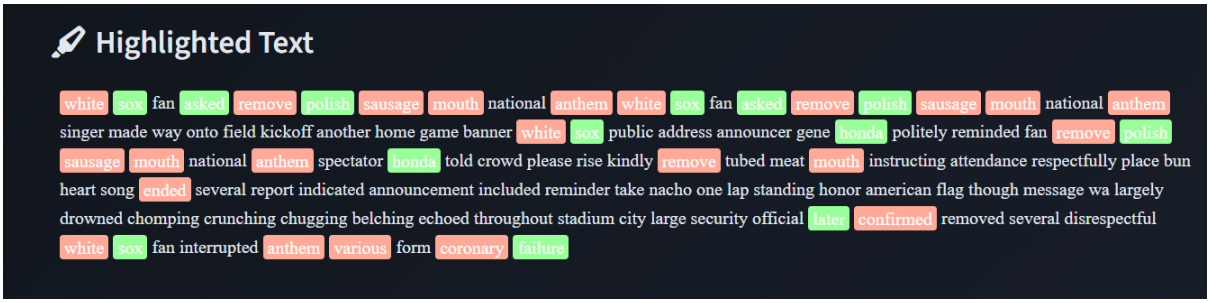

Figure 14: Influential Words Highlighted

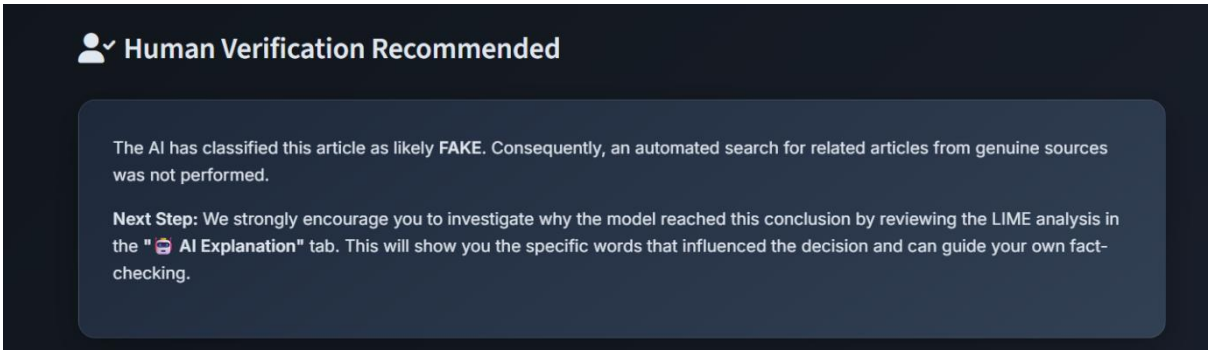

Figure 15: Verification Sources Tab

- Test Status: Passed ☑
    - **Reason:** The model performed as expected, correctly identifying the article as 'FAKE' with 73.8% confidence. The LIME analysis (Figure 6) reveals that the model's decision was influenced by the unusual and sensational combination of words like 'anthem,' 'sausage,' 'coronary,' and 'failure', which are statistically uncommon in legitimate news articles. This demonstrates the model's ability to look beyond simple keywords and analyze contextual patterns.

Test Case 1.3: Evidence Retrieval for Legitimate Articles

- **Objective:** To confirm that the system retrieves relevant articles
- **Test Steps:**
    - Retrieved from “https://www.washingtontimes.com/news/2025/jul/17/white-house-says-no-special-counsel-coming-epstein-case/”
    - Title: “White House says no special counsel coming in Epstein case”
    - Text: Input the complete article content as retrieved from the source URL
    - Click the **"Analyze Article"** button.
- **Expected Outcome:**
    - Prediction: **REAL** (with confidence >70%)
    - Verification Sources: Articles from reputable sources should be retrieved.
    - Summary: A concise summary should be generated.
- **Observed Outcome:**

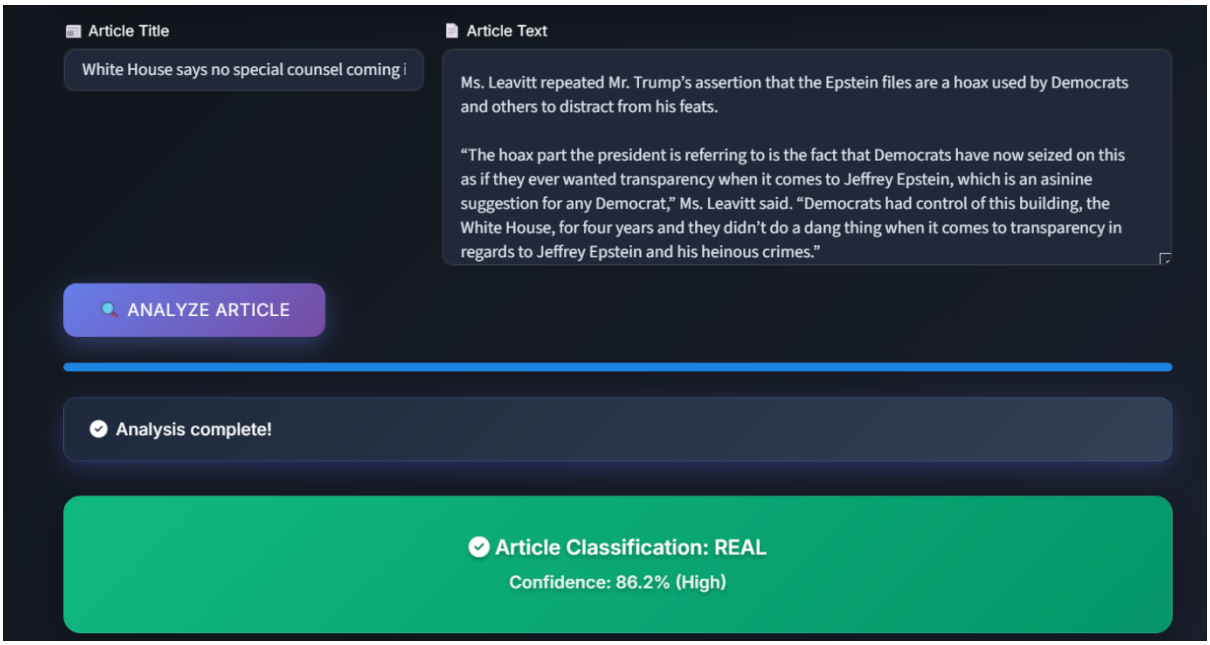

Figure 16: Real/Fake News Classification

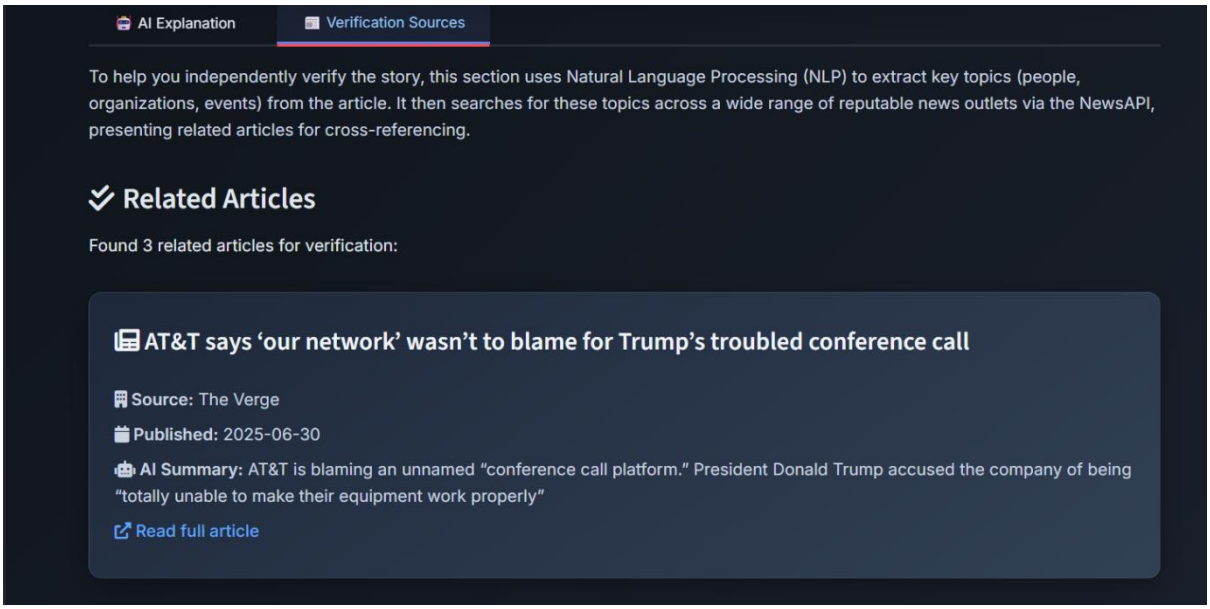

Figure 17: Verification Sources Tab with Article One

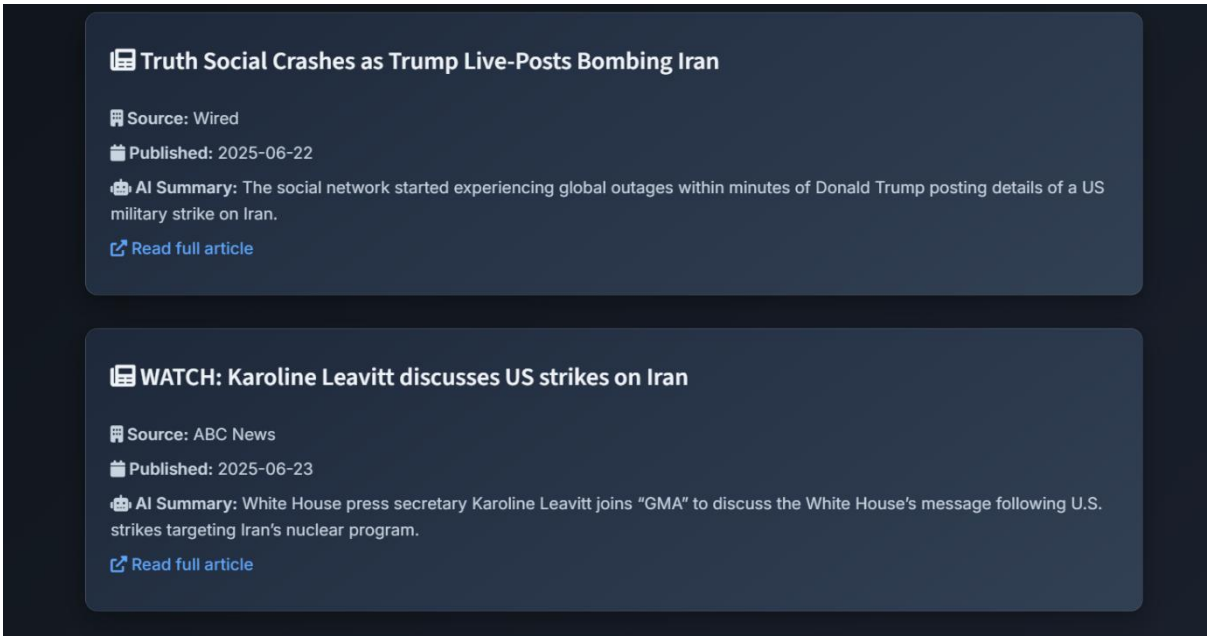

Figure 18: Verification Sources Tab with Article Two and Three

- **Test Status: Partially Passed**
    - **Reason:** The articles retrieved are somewhat relevant to the pasted article. Achieving full accuracy is impossible while maintaining good inference speed, which is a known limitation. However, the fact that the retrieved articles are

similar and include concise summaries indicates that this test case is a partial success.

II. Edge Case and Input Validation Testing:

Test Case 2.1: Very Short Article

- **Objective:** To test system handling of articles too short for analysis.
- **Test Steps:**
  - **Title:** “Did they post their votes for Hillary already?”
  - **Text:** “Hillary votes might be posted”
  - Click the **"Analyze Article"** button.
- **Expected Outcome:** Warning about short text
- **Observed Outcome:**

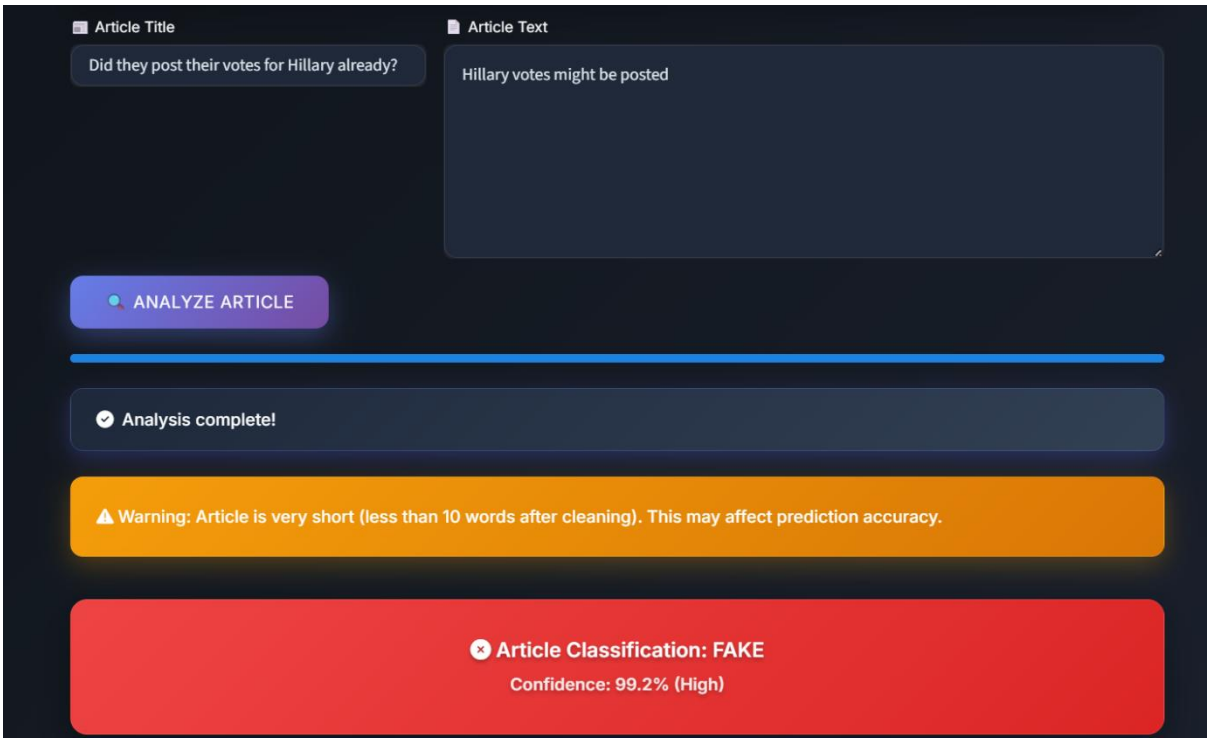

Figure 19: Real/Fake News Classification

Figure 20: Bar chart of Influential Words

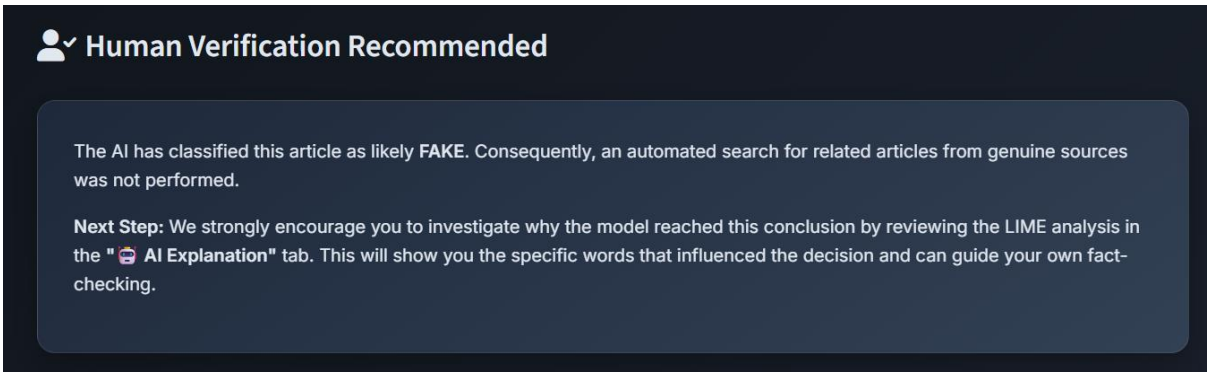

Figure 21: Verification Sources Tab

- **Test Status: Passed** ☑
    - **Reason:** The system correctly issued a warning that the article text is very short. This is a crucial validation step, as the AI model was trained on a dataset where articles with fewer than 10 words were excluded during preprocessing. This warning prevents the user from over-relying on a prediction made from insufficient data, demonstrating reliable system design.

Test Case 2.2: Empty Input

- **Objective:** To ensure proper handling of empty input fields.
- **Test Steps:**
  - Title and Text fields should be empty
- **Expected Outcome:**
  - Display error message prompting user to fill both fields.
  - No analysis should be performed
- **Observed Outcome:**

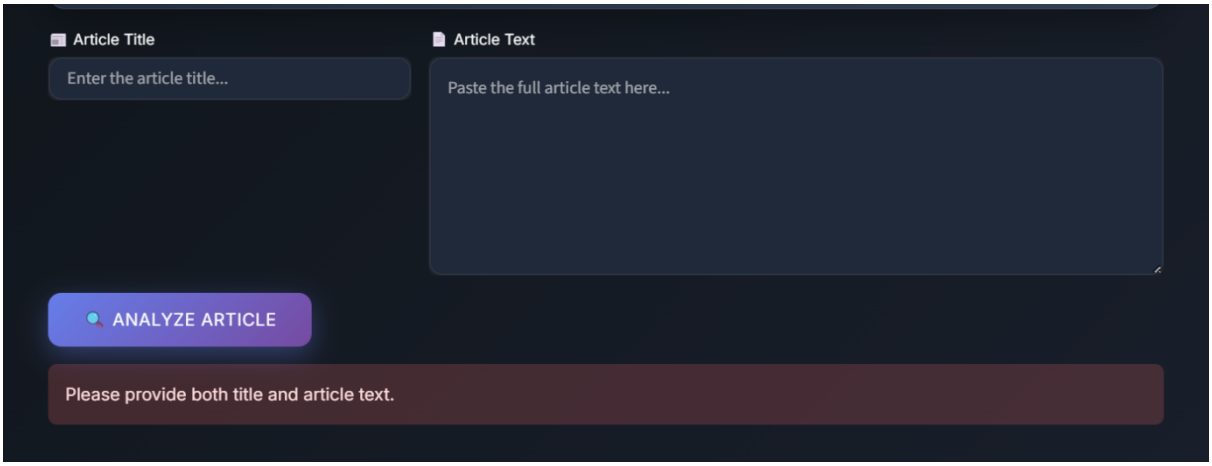

Figure 22:Warning Message for Empty Fields

Test Case 2.3: Non-English Input

- **Expected Outcome:** Display error message stating only English content is supported.
- **Test Steps:**
  - **Title:** 标题：

    中国成功发射新一代气象卫星，提升全球气象监测能**力**
  - **Text:** 正文：

    北京时间**2025**年**7**月**18**日**9**时**12**分，**中国在西昌**卫星发射中心成功发射了新一代静止轨道气象卫星“风云四号**E**星 **此次**发射由长征三号乙运载火箭执行，卫星顺利进

入预定轨道，任务取得圆满成功 据中国国家空间局介绍，“风云四号E星”具备更高分辨率的遥感成像能力和更灵敏的探测仪器，将显著提升我国在灾害预警、农业气象、环境监测等领域的服务水平。该卫星还将加强“一带一路”沿线国家的气象合作，为全球气候变化研究提供数据支持 专家指出，此次成功发射标志着我国气象卫星技术进入世界先进水平，将进一步增强我国在全球气象观测系统中的话语权和贡献能力。

- **Observed Outcome:**

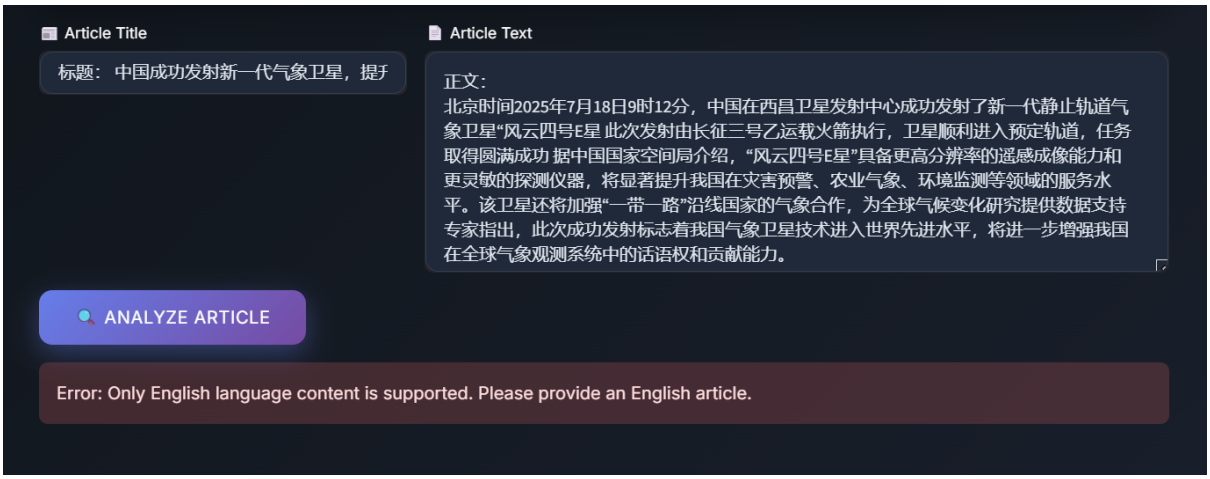

Figure 23: Warning Message about Non-English Content

- **Test Status: Passed** ☑
    - **Reason:** The system identified non-English content and informed the user that only English content is supported.

Test Case 2.4: Mixed Signals

- **Objective:** To examine how the model responds to a complex article that combines true information with false claims, which is a common disinformation tactic.
- **Test Steps:**
    - **Title:** "White House Confirms UFO Debris in Roswell, Scientists Link it to New Energy Source"
    - **Text:** "WASHINGTON D.C. — In a press conference today, White House officials confirmed long-standing rumors about the 1947 Roswell incident, stating that

debris recovered was indeed of extraterrestrial origin. Records from the event, now declassified, describe the material as 'not of this Earth.' (This part is fabricated). This announcement comes as a team of physicists at MIT, led by Dr. Evelyn Reed, published a paper in the journal *Nature* detailing a new method for creating stable fusion reactions at lower temperatures. The paper, titled 'A Pathway to Cold Fusion,' has been peer-reviewed and is seen as a significant breakthrough in energy research. (This part is based on a real-sounding, plausible event). Anonymous sources are now claiming the government is attempting to connect the Roswell technology to Dr. Reed's research."

- Click the "Analyze Article" button.

- **Expected Outcome:**
  - The model should classify the article as FAKE due to the highly exaggerated and false claims.
  - The primary goal is to examine the LIME (AI Explanation) chart to see which signals the model prioritized. Did it weigh the sensational "UFO Debris" language more heavily than the grounded, scientific language of the "MIT research"?

- **Observed Outcome:**

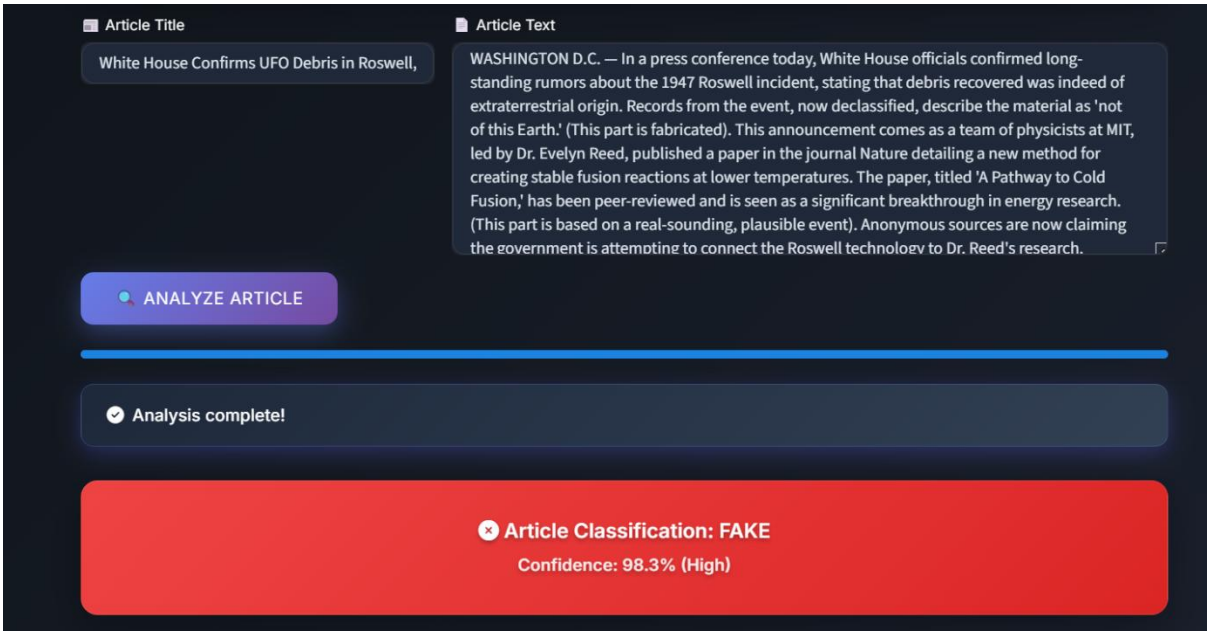

Figure 24: Fake Classification Card

Figure 25: Bar chart of Influential Words

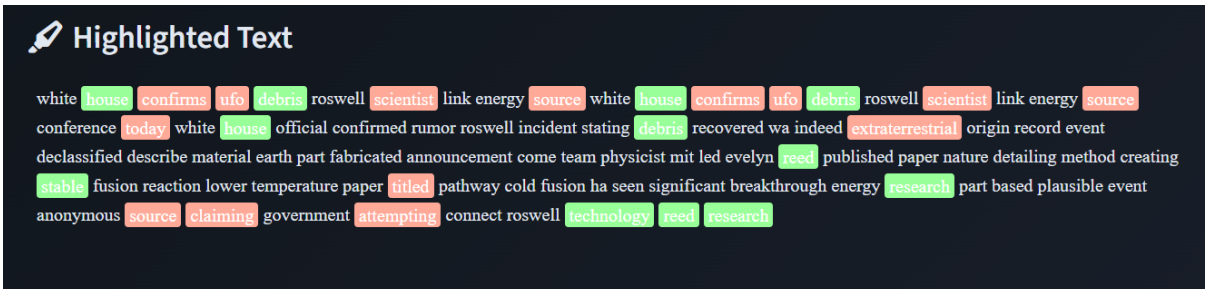

Figure 26: Influential Words Highlighted

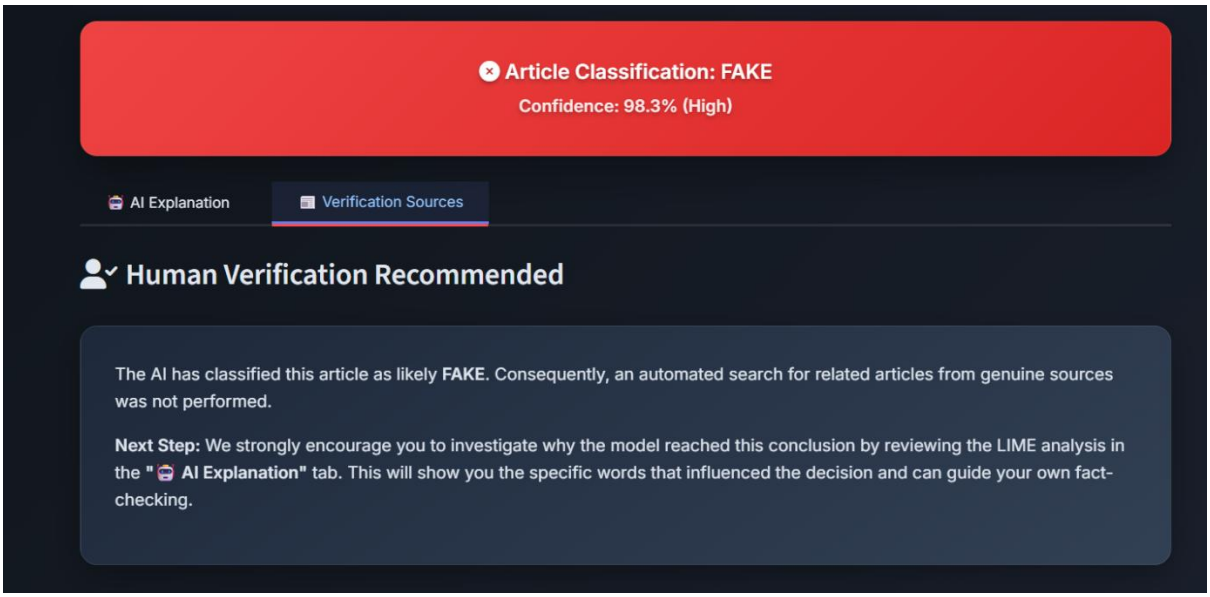

Figure 10: Verification Sources Tab

- **Test Status:** Passed ☑
- **Reason / Analysis:** The LIME analysis (Figure 18) shows that the model correctly labeled the article as 'FAKE' by focusing on the sensational and false parts of the text. The main factor in its decision was the word 'ufo', which had the largest negative impact score of -0.335. This was supported by other keywords linked to conspiracy or false claims, such as 'claiming,' 'source,' and 'extraterrestrial'. Interestingly, the model also recognized more credible parts of the article, giving positive (REAL news) scores to words like 'reed' (the scientist's name), 'research,' and 'technology'. This shows the model was able to understand both the false and believable parts of the article. In the end, it gave more weight to the clear signs of disinformation than to the factual-sounding language. This test confirms the model's strength in handling disinformation that mixes truth with lies.