

FAKE NEWS DETECTION USING HEURISTIC ALGORITHM

G. Vimalraj*, M. Sukil**, Mrs. M. Sharon Nisha***

*(Computer Science and Engineering, Francis Xavier Engineering college, Tirunelveli

Email: vimalrajg.ug22.cs@francisxavier.ac.in)

** (Computer Science and Engineering, Francis Xavier Engineering college, Tirunelveli

Email: sukilm.ug22.cs@francisxavier.ac.in)

*** (Assistant Professor Computer Science and Engineering, Francis Xavier Engineering college, Tirunelveli

Email: sharonnisha@francisxavier.ac.in)

Abstract:

The rapid dissemination of misinformation and disinformation on digital platforms has escalated the global issue of fake news. This paper presents the development of a Fake News Analyzer, a web-based application designed to assist users in validating the authenticity of news articles. The system integrates a heuristic algorithm with real-time Google search analysis to evaluate the credibility of submitted news content. Users can enter any news text, which is then compared against trusted sources such as The Hindu, The Hindu Tamil, Dinamani, Dinakaran, Vikatan, and international fact-checking databases. The backend is built using Flask, while the frontend leverages React for an interactive user experience. The system employs a token-based authentication mechanism using JWT to ensure secure access and user history tracking. Each analyzed article is stored along with its classification (Real or Fake) and timestamp, allowing users to export their analysis history. The heuristic algorithm assigns credibility scores based on the frequency of matches with reliable sources, contextual relevance, and the presence of specific keywords.

Keywords: Fake News Detection, Heuristic Algorithm, Flask, React, Misinformation, JWT Authentication, Credibility Score, Real-Time Analysis, Trusted Sources, News Verification, Google Search Integration, Web Application, User History Tracking, Disinformation, Natural Language Processing (NLP).

I. Introduction

Information moves quickly through a variety of venues in today's digital ecosystem, such as blogs, instant messaging apps, social media, and online news portals. While this metamorphosis has made global communication more accessible, it has also led to an unknown swell in the distribution of misinformation, frequently in the form of fake news. Fake news refers to fabricated information that mimics licit news in form but lacks the veracity and intent of genuine journalism. This miracle can beget serious consequences ranging from public fear to political uneasiness, fiscal fraud, and corrosion of trust in institutions. Accordingly, automating the process has come an

active area of exploration within the fields of natural language processing(NLP), information reclamation, and artificial intelligence(AI). numerous machine literacy- grounded approaches have surfaced, using labeled datasets and complex models like BERT, RoBERTa, and LSTM networks to classify news papers grounded on verbal patterns and contextual meaning. still, the need for real- time, featherlight, and stoner-friendly verification systems has grown more important, especially for the general public who warrant specialized moxie. Our design — Fake News Analyzer — islands this gap by espousing a heuristic- grounded algorithm that mimics mortal geste

validating news by searching for its presence in estimable sources using Google Hunt. The design is enforced as a full- mound web operation, using Flask for the backend and Reply for the frontend. It includes features similar as stoner authentication with JWT, live news credibility analysis, storehouse of analysis history, and import options. This ensures druggies ca n't only corroborate suspicious content in real-time but also keep track of their former queries and results for reference. likewise, the system is designed to be extendable — allowing future integration with fact- checking APIs like Google Fact Check, PolitiFact, and Snopes. The current perpetration prioritizes translucency and ease of use, aiming to empower druggies in developing critical digital knowledge chops essential for navigating moment's information-rich yet trust-deficient online terrain.

II. Literature Survey

The surge of fake news has prompted researchers and technologists to develop various computational models to detect and mitigate its spread. This literature survey explores notable contributions in the domain of fake news detection, highlighting their methodologies, strengths, and limitations.

2.1 Traditional Machine Learning Approaches

Early efforts in fake news detection relied on supervised learning algorithms such as Naïve Bayes, Support Vector Machines (SVM), and Random Forests. These models primarily used linguistic features like word frequency, term frequency-inverse document frequency (TF-IDF), sentiment analysis, and stylistic cues. For instance, Potthast et al. (2017) proposed a style-based classifier that identified fake news using features such as readability and sentence structure. Although these models were interpretable and computationally efficient, they lacked context understanding and were highly dependent on dataset quality.

2.2 Deep Learning Models

To improve contextual understanding, deep learning models were introduced. Techniques such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) proved

effective in capturing sequential dependencies in text data. Researchers also used Word2Vec and GloVe embeddings to encode semantic meaning.

2.3 Transformer-based Architectures

With the advent of transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa, fake news detection saw significant improvements. These models, pre-trained on massive corpora, could understand the nuanced context of words in relation to their surroundings. Works by Devlin et al. (2019) demonstrated that fine-tuned BERT models outperformed traditional and RNN-based methods on benchmark datasets like LIAR and FakeNewsNet. Despite their accuracy, these models were often too resource-intensive for lightweight real-time applications.

2.4 Source-Based and Fact-Checking Methods

Rather than analyzing the content of news articles, some systems focused on verifying the credibility of sources. (2020) introduced NewsGuard and other APIs that assign credibility scores to publishers. Additionally, platforms like Snopes, PolitiFact, and Google Fact Check API provide metadata and human-verified labels for fact-checked stories. These systems are highly accurate but limited by the coverage of known stories.

2.5 Heuristic and Hybrid Systems

Hybrid models combine content analysis with external validation techniques. Our approach falls under this category. We use a heuristic-based algorithm that checks the presence of the news across trusted web sources using Google Search. This mimics human verification behavior while maintaining low computational complexity. Unlike models that rely entirely on pre-labeled datasets, our system dynamically verifies claims in real time, making it more adaptable to emerging fake news.

III. System Architecture / Methodology

The Fake News Analyzer is developed using a modular and scalable architecture that integrates natural language processing (NLP), machine learning (ML), and heuristic-based verification to

effectively analyze and authenticate news content. The architecture is composed of four major components: the frontend, backend, database, and external services (search APIs and trusted sources). The system workflow begins with the user input layer, where a user submits a news article or headline through the frontend interface.

Users are required to authenticate via JWT-based login or signup to access the analyzer and history features. Once the input is received, it undergoes text preprocessing to clean the content by removing stopwords and punctuation. The heuristic verifier then sends the processed text to a search engine API (like Google Search), and the results are filtered against a database of trusted news sources such as The Hindu, Reuters, BBC, PolitiFact, and Snopes.

Simultaneously, an optional transformer-based ML model (BERT fine-tuned on the LIAR dataset) predicts whether the news is “Real” or “Fake” with an accuracy of approximately 92%. The result aggregator combines the ML prediction with the heuristic credibility score and presents the final verdict along with relevant source links. The history logger stores each analyzed input, result, timestamp, and source URLs into a SQLite database, and users are given the option to export this data as a CSV file. The frontend, developed in React.js, features a responsive user interface for login, analysis, and history viewing, and communicates with the Flask-based backend using Axios.

The backend manages routing, authentication, and coordination between the ML model, database, and external APIs. Security is ensured through JWT authentication, password hashing, and input validation to prevent injection attacks. The overall methodology provides a lightweight yet powerful solution, offering real-time verification, multilingual support (with trusted sources), and high transparency through source-linked results, making it suitable for both local and cloud deployment.

iv. Algorithm and Model

The Fake News Analyzer employs a hybrid approach that combines a pre-trained Machine

Learning (ML) model with a Heuristic-based verification algorithm. This dual strategy enhances the accuracy and trustworthiness of the results, especially in real-time user scenarios.

The Machine Learning Model is based on a transformer-based NLP architecture (e.g., BERT) and utilizes the Hugging Face Transformers library with a PyTorch or TensorFlow backend. The model is trained on the LIAR dataset, a benchmark collection of 12,836 labeled statements, categorized as true, mostly true, half true, mostly false, false, or pants on fire. Supplementary datasets such as FakeNewsNet and Kaggle fake news datasets are also used.

The training process involves data preprocessing steps like tokenization, padding, truncation, lowercasing, and stopword removal. BERT is then fine-tuned using a binary classification approach (Real vs Fake) with AdamW optimizer, a learning rate of $\sim 2e-5$, and 3-5 epochs. To enhance the model, a heuristic-based algorithm is also implemented.

This algorithm validates the credibility of news by performing a Google or Bing search for the input text and extracting the top results (title + URL). These results are then compared to a list of trusted sources like The Hindu, Reuters, BBC, Snopes, and PolitiFact. Each match from a trusted source contributes +1 to a credibility score. If the majority of top results come from credible sources, the news is classified as Real News. If not, it is classified as Fake News. Finally, the results from both the ML model and the heuristic algorithm are merged; if they agree, the news is confirmed, but if they disagree, a warning is issued along with supporting source evidence.

v. Implementation

The implementation of the Fake News Analyzer is structured as a full-stack web application using a modular and scalable architecture.

1. Frontend (React.js)

The Frontend of the application is built using React.js with React Router for navigation. It includes several key features such as Login & Signup functionality with JWT-based

authentication, allowing users to securely log in and access their personalized content. The News input analyzer page enables users to enter news text and dynamically renders the results from both the ML model and the heuristic algorithm, displaying the analysis with relevant source links.

A History page is included to show the user's query history, complete with timestamps and filtering options. Users can also export their history as a CSV file for easy reference. The UI is designed to be simple yet visually engaging, with normal CSS and React transitions used for animations. Key components of the frontend include LoginPage.js, which handles the login and signup process; AnalyzePage.js, where users can input text and view the analysis results; and HistoryPage.js, which displays the user's previous analyses and allows for filtering of past queries.

2. Backend (Flask + Python)

The Backend of the application is built using Flask, leveraging several essential libraries such as Flask-JWT-Extended for JWT-based authentication, SQLAlchemy for database management, Scikit-learn for machine learning functionalities, and Transformers for NLP model integration. The backend also uses Requests and BeautifulSoup for web scraping. Key endpoints include /signup for registering new users, /login for generating a JWT token, /analyze for processing news input and running the ML + heuristic model to return results, and /history for retrieving and displaying the stored user query history.

Simultaneously, a web search is triggered to gather relevant URLs and titles from trusted sources. The credibility scoring system evaluates these sources, and the results from both the ML model and the heuristic algorithm are combined to determine the final output, whether the news is classified as real or fake. This workflow ensures a robust and accurate analysis of news credibility.

3. Machine Learning Model Integration

The model used for news classification is a fine-tuned BERT model designed for binary classification, determining whether news is Fake or Real. The model is loaded during Flask startup

to ensure it's ready for real-time predictions. Preprocessing of the input text is handled using the Hugging Face Tokenizer, which tokenizes the text before passing it into the model. After the model makes a prediction, postprocessing is performed by mapping the prediction labels to user-friendly categories: "Real News" or "Fake News", ensuring easy interpretation of the results.

4. Database (SQLite)

- Users: This table stores the username and hashed passwords for each registered user to ensure secure authentication.
- History: This table keeps a record of each user query, the result (whether the news is real or fake), and the timestamp of the query.

The backend functionality allows for tracking individual user activity, ensuring that each user's queries are stored and accessible. Additionally, it enables search, filter, and export capabilities in the frontend, providing users with the ability to review and manage their past analyses easily.

5. Security and Token Handling

Tokens are stored in the local storage of the user's browser to maintain session persistence. Authentication headers are added to every request to ensure secure access to the protected endpoints.

For password security, passwords are hashed before being stored in the database using werkzeug.security to protect user data from unauthorized access.

6. Deployment

The development tools used for this project include Visual Studio Code for code editing, Postman for API testing, and GitHub for version control and collaboration.

In the future, the project plans to scale through Docker containerization and host it on platforms like Heroku or Render for easier deployment and scalability.

VI. Result and Analysis

The Fake News Analyzer was tested using a variety of real-world news samples, both genuine and fabricated, to assess the model's accuracy,

reliability, and usability. The results were analyzed from two perspectives: machine learning model performance and heuristic credibility assessment.

1. Machine Learning Model Performance

The model used is a fine-tuned BERT (Bidirectional Encoder Representations from Transformers), which is a state-of-the-art transformer-based architecture for natural language processing. The model is trained on the LIAR dataset, which contains labeled statements, and is augmented with additional samples from Kaggle's fake news datasets to improve its generalization.

The model's performance:

- Accuracy: 92%
- Precision: 90%
- Recall: 89%
- F1 Score: 89.5%

These metrics indicate strong performance in classifying news as real or fake.

2. Heuristic Credibility Scoring

The system also performs a Google-based real-time web search to find similar news articles and checks if the URLs belong to a set of trusted news sources, such as The Hindu, BBC, NDTV, The Hindu Tamil, and Vikatan.

A credibility score is determined by taking into account a number of factors:

- The number of matching sources found from trusted domains
- The domain authority of the sources

If more credible sources are found, the system classifies the news as Real News. If no credible matches are found or if misleading sources are identified, the system classifies the news as Fake News.

3. User Feedback and Usability

The User Interface (UI) received positive feedback from test users, who appreciated its clean design, smooth animated transitions, and responsive feedback. The interface was praised for being intuitive and visually appealing, enhancing the overall user experience.

In terms of functionality, users found the history tracking feature particularly useful for keeping a record of their past analyses. The ability to export history as a CSV file was also highly valued, especially for academic and research tracking, as

it allows users to easily organize and analyze their past queries and results.

4. Limitations Observed

- Some satirical content was occasionally marked real due to formal writing.
- Heuristic checks may fail if internet access is restricted or source domains are newly created.

5. Improvement Scope

- Integrating additional fact-checking APIs like Google Fact Check or Snopes.
- Training on multilingual datasets for better regional fake news detection.
- Refining heuristics using NLP-based text similarity instead of domain matching alone.

VII. Conclusion

The Fake News Analyzer project successfully integrates machine learning with real-time heuristic validation to combat the growing issue of misinformation. By utilizing a fine-tuned BERT model trained on reliable datasets, the system demonstrates strong accuracy in identifying fake versus real news articles based on their content. This is further strengthened by a heuristic credibility scoring mechanism, which evaluates the trustworthiness of news based on live web search results and domain credibility.

The hybrid approach not only improves classification reliability but also enhances transparency by showing users the supporting sources. With an intuitive user interface, features like search history, result filtering, and CSV export make the system practical for both casual users and academic research.

While the system performs well across a wide range of test cases, it is not without limitations. Satirical content, newly emerging fake news without digital traces, and internet-dependent validation are areas that pose challenges. Nonetheless, this platform offers a strong foundation for scalable fake news detection and can be improved further by incorporating additional fact-checking APIs, regional language support, and more sophisticated NLP techniques.

Overall, the project makes a meaningful contribution to digital literacy and the fight against disinformation, offering a useful tool for

individuals, educators, journalists, and policymakers.

VIII. Future Scope

The *Fake News Analyzer* provides a strong base for misinformation detection, and several enhancements can be made in the future to baden its capabilities and impact:

1. Multilingual Support: Extend the analyzer to support regional languages like Tamil, Hindi, etc., to target a wider demographic affected by local fake news.
2. Advanced Fact-Checking APIs: Integrate external APIs such as Google Fact Check, Snopes, or Politifact to cross-reference claims with verified sources.
3. Image and Video Analysis: Incorporate computer vision models to analyze the credibility of visual content often used in spreading misinformation.
4. Real-time Browser Extension: Develop a Chrome/Edge extension that scans news articles in real time while browsing and flags potential misinformation instantly.
5. Social Media Integration: Extend the tool to detect fake news circulating on platforms like Twitter, Facebook, and WhatsApp using social scraping and NLP.
6. Crowdsourced Reporting: Allow users to flag and submit suspicious news for analysis, improving dataset diversity and enabling community-driven detection.
7. Sentiment and Emotion Analysis: Add modules to understand emotional triggers in fake news, which often uses sensationalism to go viral.
8. Adaptive Learning: Use continual learning techniques to keep the ML model updated with the latest patterns in fake news tactics.
9. Backend Enhancements: Deploy scalable cloud-based infrastructure (e.g., AWS, GCP) to serve large volumes of requests and provide real-time response.
10. Explainable AI: Integrate explainability tools like LIME or SHAP to provide users with insights on why a certain article was flagged as fake or real.

IX. References

1. Sadia Afroz, Michael Brennan, and Rachel Greenstadt. Detecting hoaxes, frauds, and deception in writing style online. In ISSP'12.
2. [2] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research, 2017.
3. [3] Solomon E. Asch and H. Guetzkow. Effects of group pressure upon the modification and distortion of judgments. *Groups, leadership, and men*, pages 222--236, 1951.
4. [4] Meital Balmas. When fake news becomes real: Combined exposure to multiple news sources and political attitudes of inefficacy, alienation, and cynicism. *Communication Research*, 41(3):430--454, 2014.
5. [5] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In IJCAI'07.
6. [6] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. *First Monday*, 21(11), 2016.
7. [7] Prakhar Biyani, Kostas Tsioutsoulis, and John Blackmer. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In AAI'16.
8. [8] Jonas Nygaard Blom and Kenneth Reinecke Hansen. Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*, 76:87--100, 2015.
9. [9] Paul R Brewer, Dannagal Goldthwaite Young, and Michelle Morreale. The impact of real news about fake news: Intertextual processes and political satire. *International Journal of Public Opinion Research*, 25(3):323--343, 2013.
10. [10] Carlos Castillo, Mohammed El-Haddad, Jürgen Pfeffer, and Matt Stempeck. Characterizing the life cycle of online news stories using social media reactions. In CSCW'14.