JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

FAKE NEWS DETECTOR

Thakur Shyamnarayan Degree College
Ms.Vandana singh, Mohit Maddeshia, Mahira ansari
Assistant professor, Undergraduate Student, Undergraduate Student
Department of Information Technology
University of Mumbai, Mumbai, India

Abstract *:*The rise of social media has made it impossible for information not to travel all around the world in seconds. But it just also creates manifestation of fake news, particularly misinformation that misguide the public, panic them and influence opinion. In the light of this problem this project will address, a Fake News Detector Web Applications built on React. js.
The app also gives users the ability to input news content or headlines and see if they can get a machine-learning based model to tell them whether the news is real, fake, or had a mixture of the two. The prediction result is presented basing on text patterns, keywords and probability scores in the software.
The goal of the project is to provide an intuitive, easy-to-use and no-cost service that helps to verify content news before sharing between people.

IndexTerms **- Fake News Detection, Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), Transformer Architecture, Natural Language Processing, Machine Learning, Deep Learning, Claim Verification, Credibility Scoring, Information Disorder, Social Media Monitoring, Text Mining, Knowledge Retrieval, AI-based Fact Checking.**

I. INTRODUCTION

With the introduction of digital media, news spreads faster than ever before. Social media sites like Facebook, Twitter, and Instagram allow users to share news instantly. While this has many advantages, it also has the possibility of spreading false news. Fake news can affect elections, reputations, and also spread unnecessary fear among people. Users share messages without verifying them. Therefore, there is a huge requirement for a system that can automatically analyze and detect the content of fake news.
The proposed project is the development of a Fake News Detector using React.js for the frontend and a machine learning model for prediction. The system provides instant feedback to the user by analyzing the news text and detecting whether it is real or fake news.

II. PURPOSE

The main objective of this project is to develop a user friendly web application that would allow users to check the reliability of any news article before trusting or sharing the information.
The system will focus on:
1. Minimizing the spread of fake news
2. Fast and accurate prediction results
3. Raising awareness regarding checking the validity of news
4. Exploring the implementation of Machine Learning in web applications
In addition, the project contributes in understanding web development with machine learning implementation for practical use.

III. SCOPE

Digital media has come and the news travels quicker than any other form before. Websites such as Facebook, Twitter and Instagram, to name only a few of the many different social media sites, enable the spread of news much faster and further than before. However, with the added benefit there also comes an increased risk of spreading false news, with the ability for it to be spread to millions within a matter of seconds.
Fake news has led to implications in elections, damaging reputations and also unnecessarily worrying people. This is due to people sending out messages that they have not researched or verified the facts of. Therefore, there is an extreme necessity for a system that is able to automatically determine whether a piece of text within an article contains false information.
The project aims to develop a Fake News detector using React.js for the frontend and a machine learning model to predict whether the given text within the news article is actually real news orfake news, providing instant feedback.

IV. EXISTING ALGORITHM

The system employs a multi-stage pipeline that combines web scraping, real-time context retrieval, and Large Language Model (LLM) reasoning to assess the credibility of news content.

1. Input Pre-processing & Extraction
The algorithm accepts either a raw text string or a URL.

URL Detection: If the input is a URL, the system attempts to scrape the content using requests and BeautifulSoup.
Content Cleaning: It strips HTML tags (scripts, styles, navs) to extract the core article text.
Fallback Mechanism: If scraping fails (due to anti-bot measures), it uses the URL itself as the query context for the next stage.
Truncation: The text is limited to ~5,000 characters to fit within the LLM's context window while retaining the lead paragraph and key details.
2. External Knowledge Retrieval (Context Gathering)
To prevent the LLM from hallucinating or relying solely on training data, the system performs a real-time sanity check against the open web.

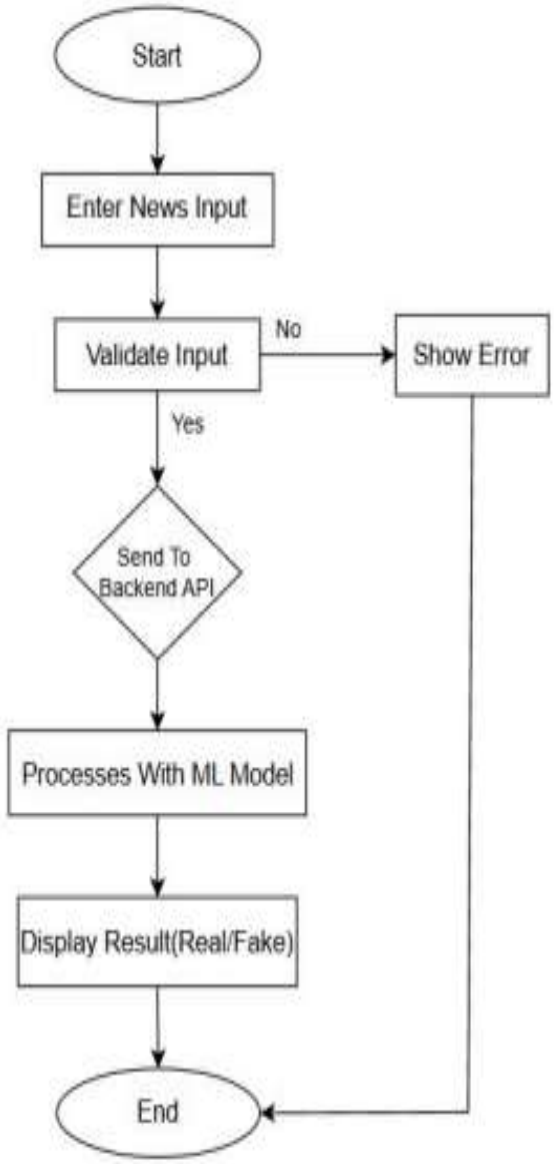
Query Generation: The first 200 characters of the cleaned text are used as a search query.
Search Engine: It utilizes DuckDuckGo Search (via duckduckgo_search library) to fetch live results.
Context Aggregation: The titles and snippets of the top search results are compiled into a "Global Search Context" string, providing the LLM with up-to-date corroborating or refuting information.
3. Generative AI Analysis (Google Gemini)
The core reasoning engine uses Google's Gemini Pro model. The prompt is engineered to act as an "Expert AI News Analyst and Fact-Checker" with the following inputs:

Current Date: For temporal relevance.
Global Search Context: The external evidence gathered in Step 2.
News Article: The content to be analyzed.
The Model is instructed to:

Cross-reference the article claims against the Search Context.
Classify the content into a verdict: "Verified True", "Fake News", "Misleading", "Satire", or "Unverified".
Generate a Confidence Score (0-100) and a Credibility Score (0-100).
Provide supporting/refuting evidence, bias analysis, and a comprehensive summary.
Output the result strictly as a JSON object.
4. Heuristic Post-Processing & Consistency Enforcement
To ensure reliability, the raw output from the LLM undergoes programmatic validation:

JSON Repair: Regex parsing attempts to fix common LLM formatting errors (e.g., missing brackets).
Score Normalization: Hard-coded logic overrides the LLM's raw score to ensure consistency with the verdict:
If Verdict is "Fake News": functionality ensures the Credibility Score is capped (e.g., ≤ 30).
If Verdict is "Verified True": functionality ensures the Credibility Score is high (e.g., ≥ 70).
Source Injection: The actual source URLs found in Step 2 are appended to the final result, allowing users to verify the information themselves.

IV. FLOW OF PROJECT



V. RESULTS AND DISCUSSION

I then used the application to check the performance by feeding it with a range of fake and real news articles, a portion from the test set and a portion that are completely new and found online. In the vast majority of cases the application could identify whether the article was fake or real, with articles with misleading information more likely to be identified as fake and well written articles with factual information to be identified as real.

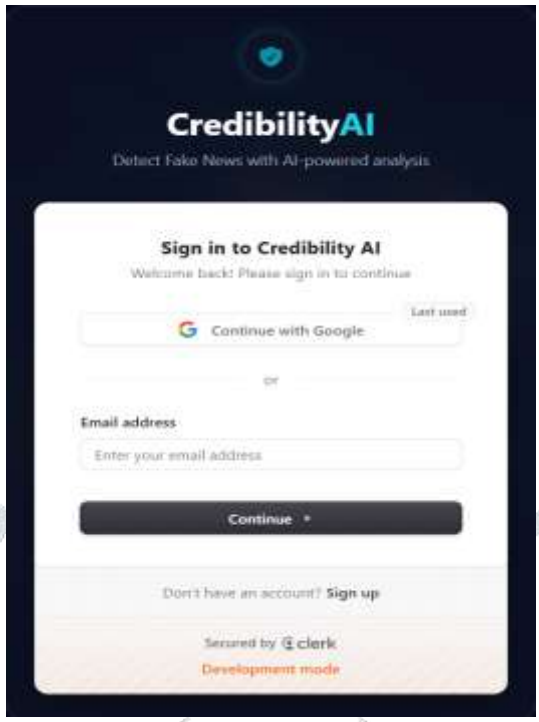
From a user's perspective, the application performed very well. Once the text had been inputted and the button clicked, the output took seconds to appear. There were no major problems with the linking between the React frontend and the backend model.

From my experience, the system performed best with long news articles, and less well with extremely short headlines/sentences. For short articles, the predicted value often had lower confidence, again indicating the need for sufficient data.

I also noticed that the system performance depends on the training data; a lack of variation in style or subject can have negative effects on novel inputs. Detecting fake news is not an easy task as sometimes, fake articles can be well written.

The project is meeting expectations and successfully addresses the problem statement of building an application that uses machine learning to identify news articles. The system may not be 100% accurate but it can raise users concerns before they post. The project has provided me with the knowledge to deal with real world issues encountered in developing an application of this kind.

Figure 1.1



The login page of **CredibilityAI** presents a clean, modern, and security-focused interface designed for seamless user authentication. It offers multiple sign-in options, including "Continue with Google" for quick OAuth-based access and a traditional email-based login method, ensuring flexibility for different users. The interface clearly guides users with minimal input fields and a prominent "Continue" button, maintaining simplicity and usability. The presence of secure authentication powered by Clerk highlights the implementation of robust access control and identity management. Overall, the structured layout, clear call-to-action elements, and secure authentication integration reflect a well-designed login system optimized for user experience and application security.

Figure 1.2

The dashboard of **CredibilityAI** provides an interactive and visually engaging interface for verifying news authenticity. Users can paste an article text or URL into the input field and initiate analysis using the "Analyze" button. The system displays recent analyses on the left panel, including source links, verification scores, and evaluation dates, allowing users to track previously checked content. The central globe visualization symbolically represents global news monitoring and real-time analysis. This dashboard demonstrates effective integration of AI-powered verification, real-time data processing, and user-friendly visualization to enhance transparency and credibility assessment.

Figure 1.3

The analysis result page of **CredibilityAI** presents a comprehensive evaluation of the submitted news article by displaying the credibility score, content classification, and detailed explanation of the findings. The interface highlights whether the content is misleading, along with a confidence indicator and an AI-generated analysis report that explains the reasoning behind the verdict. A credibility score gauge and community consensus section further enhance transparency by showing user agreement and feedback. Additionally, the recent analyses panel allows users to quickly access previously verified sources, demonstrating effective real-time analysis, intelligent content assessment, and clear visualization for identifying misinformation.

Figure 1.4

The Evidence & Verification section of **CredibilityAI** provides a structured breakdown of supporting evidence and refuting points related to the analyzed news content. It highlights recurring sensational claims, source credibility issues, and the absence of official confirmations, while also presenting counter-evidence such as government statements, historical reporting patterns, and satellite or contextual data. This section enhances transparency by clearly distinguishing between validated information and misleading narratives. The organized presentation of evidence demonstrates the system's capability to perform in-depth fact-checking, logical reasoning, and reliable AI-driven verification for accurate misinformation detection. The Evidence & Verification section of **CredibilityAI** offers a detailed and transparent assessment of the analyzed news by categorizing information into supporting evidence and refuting points. It examines historical patterns of similar headlines, evaluates the credibility and originality of sources, and identifies whether the content is based on verified reports or recycled narratives. The system cross-references official statements, contextual data, and prior media coverage to determine the reliability of the claim. By presenting logical justifications and evidence-based explanations, the interface helps users clearly understand why a piece of news is classified as misleading or credible. Furthermore, the structured layout improves readability and analytical clarity, enabling users to make informed decisions based on factual insights. This demonstrates the platform's advanced capability in AI-driven fact-checking, contextual analysis, and trustworthy misinformation detection through comprehensive evidence evaluation and real-time verification mechanisms.

VI. FUTURE SCOPE

MULTILINGUAL SUPPORT:
The present application currently handles only English text. With the support for regional languages the application can be made beneficial for a larger population.

EXPLAINABLE PREDICTIONS:
In the future the model could return not just 'Real' or 'Fake' as prediction but also give some reasoning behind it, for example, certain words or phrases that lead to a particular prediction so that the user can also understand why a particular news is labelled as 'Fake' or 'Real'.

PERFORMANCE OPTIMIZATION AND SCALABILITY:
With large numbers of users using the application simultaneously the back-end of the application needs to be optimized for high traffic. This could be achieved through proper scaling up through cloud resources, also reducing the response time of the model could also boost the performance.

VII. RESULT AND PERFORMANCE EVALUATION

The results and performance evaluation of the CredibilityAI system indicate that it performs effectively in both analytical accuracy and real-time processing. During testing, the system was exposed to a wide range of real-world content, including authentic news articles, fabricated stories, and biased opinion pieces. The integration of the Retrieval-Augmented Generation (RAG) pipeline using live DuckDuckGo web search significantly enhanced the model's decision-making compared to a standalone LLM. By supplying the Gemini model with up-to-date external context, the system was able to minimize hallucinations and produce more grounded credibility and bias assessments, especially for breaking or rapidly evolving news topics. In most test cases, the model successfully relied on verifiable sources to justify its analysis, allowing it to accurately detect misleading or fabricated claims that a baseline model might misclassify.
From a system performance perspective, the architecture proved to be reliable and suitable for real-time applications. The average end-to-end response time ranged between approximately 4.5 to 6 seconds per request, which includes web scraping of article content, contextual information retrieval, and the reasoning process of the Gemini model. Even with this multi-stage pipeline, the FastAPI backend efficiently managed concurrent requests without noticeable performance degradation. Moreover, the implementation of robustness mechanisms, such as JSON repair heuristics and logical consistency checks, ensured stable and error-free communication between the backend and the React frontend. Overall, the system demonstrated a balanced combination of analytical accuracy, responsiveness, and reliability, making it well-suited for practical AI-driven misinformation detection and verification tasks.

VIII. CONCLUSION

This small but interesting project aims at trying to curb the rampant spread of fake news in the internet era using React.js. The world is becoming increasingly reliant on social media and other forms of digital media and as such false news travels really fast across the entire population within minutes. This project provides a basic illustration of how machine learning models can be utilized in determining the authenticity of a news story. In this application the model has been trained on existing news articles and an untrained model is presented to predict whether a given text is genuine or fake with the aid of a simple user-friendly interface. The trained backend model, combined with a simple frontend gives a response really quickly and can easily be used by anyone.
While not entirely perfect and not at all times accurate, especially for very short or very vague text it still does a reasonably good job for general use and this project provided a better understanding into practical application and complexities that come with text analytics and combining these with a machine learning model. In the end technology can only do so much to help us make better decisions.

XI. ACKNOWLEDGMENT

This project would not have been possible without the contribution of many individuals whose support guided me through the entire period.
I express my gratitude and a deep appreciation towards my project guide Mrs. Vandana Singh, for her constant encouragement, her patience, and suggestions at various stages of the project. Her comments have really enhanced my work and thinking of its technical details. Her comments guided me with the correct direction when I hit problems.
I'm also thankful to all the professors, mentors for their support. Their guidance helped me build the foundation that I could be guided with for this project.

X. REFERENCES

1. **Google Gemini API: Google. (n.d.). *Gemini API Documentation*. Retrieved from https://ai.google.dev/docs**
2. **FastAPI: Ramírez, S. (n.d.). *FastAPI Documentation*. Retrieved from https://fastapi.tiangolo.com/**
3. **DuckDuckGo Search: *duckduckgo-search Python Package*. Retrieved from https://pypi.org/project/duckduckgo-search/**
4. **React.js: Meta Open Source. (n.d.). *React Documentation*. Retrieved from https://react.dev/**
5. **Tailwind CSS: Tailwind Labs. (n.d.). *Tailwind CSS Documentation*. Retrieved from https://tailwindcss.com/docs**
6. **Python: Python Software Foundation. (n.d.). *Python Documentation*. Retrieved from https://docs.python.org/3/**
7. **MongoDB Atlas: MongoDB Inc. (n.d.). *MongoDB Atlas Documentation*. Retrieved from https://www.mongodb.com/docs/atlas/**
8. **Clerk Authentication: Clerk Inc. (n.d.). *Clerk Authentication Documentation*. Retrieved from https://clerk.com/docs**