IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Fake News Detection Using NLP & Web Scraping

[1] P.Bhaskar , [2] D.Srinivasulu, [3]N.Narasimha prasad

[1]Department of CSE, [2]Department of CSE, [3]Department of CSE
[1] Department of CSE ,
[1]KLM college of engineering for women , Kadapa, India

Abstract: Fake news detection has become a significant challenge in today's digital world. The rapid spread of misinformation through social media platforms and various websites has profound consequences, influencing public opinion, shaping political landscapes, and even impacting global events. Traditional fact-checking approaches often struggle with the sheer volume and velocity of online information dissemination. To address this, automated fake news detection systems leveraging technology have emerged as a crucial area of research and development. These systems aim to automatically identify whether a news item is real or fake, offering a potential solution to ensure the trustworthiness of the information we consume, support ethical journalism, and safeguard public trust in news sources.

Index Terms **- NLP, WEB SCRAPING**

I. INTRODUCTION

The current digital transformation, the internet is the primary information source, but this accessibility is marred by the proliferation of fake news – deliberately fabricated information mimicking legitimate journalism, often intended to mislead, manipulate, or generate revenue. This misinformation has severe real-world consequences, necessitating automated detection systems.

Which presents an intelligent, web-based Fake News Detection System enabling users to input a news headline and receive an automated real/fake classification. Unlike static or keyword-based approaches, this system utilizes live data retrieval and semantic similarity analysis. It dynamically fetches relevant articles via the Google Custom Search API, extracts headlines using web scraping, and employs a Sentence-BERT model for semantic comparison between the input and fetched content. This allows for domain-agnostic, time-sensitive, and multilingual analysis, adapting to evolving news contexts.

The system features a modular and extensible architecture. A user interface (HTML/CSS) served by Flask receives headline input. The backend then uses the Google API systems. By combining technical innovation with social responsibility, this project seeks to become a meaningful contributor to the global fight against misinformation and fake news in the digital transformation.

2.PROPOSED SYSTEM

The proposed Fake News Detection System offers a dynamic and adaptable solution by employing a semantic similarity approach. When a user inputs a news headline, the system utilizes the Google Custom Search API to retrieve a set of relevant news articles from reputable sources in real-time. The titles of these articles are then extracted using web scraping techniques with BeautifulSoup. The core of the detection process involves using the Sentence-BERT model to generate vector embeddings for both the user-submitted headline and the scraped titles. The cosine similarity between these embeddings is calculated to measure their semantic relatedness. An average similarity score is then computed, and if it exceeds a predefined threshold, the input headline is classified as real; otherwise, it is labeled as potentially fake.

3. MODULES

3.1 Flask (Web Framework Module)

In this module, Flask, a lightweight Python web framework, is used to create the web application. Flask is responsible for handling HTTP requests, rendering HTML templates, and integrating the backend logic with the frontend user interface. Functionality in Our Code:

- Initializes the Flask web application (app.py).

  Renders the homepage and result page using HTML templates.

- Captures user input (query) from the input form.
- Connects with backend modules (scraping, search, and BERT) to process the query and return results.

3.2 BeautifulSoup (Web Scraping Module)

This module utilizes BeautifulSoup, a powerful Python library used for extracting data from HTML and XML files. It is employed for web scraping to gather headlines and content from various news websites.
Functionality in Our Code:

- Receives a list of URLs retrieved from the Google Search module
- Sends HTTP requests to fetch HTML content from each URL.
- Parses and extracts key HTML elements such as titles and headlines for comparison.

3.3 Sentence Transformer (BERT-Based Similarity Module)

This module uses the Sentence Transformer library, which converts sentences into dense vector embeddings using BERT (Bidirectional Encoder Representations from Transformers). It plays a critical role in determining the semantic similarity between the user-provided news headline and real news titles retrieved from the internet.
Functionality in Our Code:

- Converts the input headline and scraped article titles into numerical embeddings.
- Computes cosine similarity between vectors to determine semantic similarity.
- If the similarity score is ≥ 0.6, the headline is classified as Real News ✅; otherwise, it is labeled as Fake News ❌.

3.4 Requests (HTTP Requests Module)

The Requests library is used to send HTTP requests to external web services and retrieve content in a simple and efficient manner. It is an essential part of the system for fetching live web data.
Functionality in Our Code:

- Used in google_search.py to send API requests to the Google Custom Search API.
- Retrieves top relevant search results based on the user's input query.
- Also used in scraper.py to send requests to each URL and fetch the raw HTML content for parsing with BeautifulSoup.

3.5 Torch (Tensor Operations & Similarity Calculation Module)

This module incorporates Torch (PyTorch), a machine learning library used for high-performance tensor operations and mathematical computations. It supports the similarity analysis process by handling data in vectorized form.
Functionality in Our Code:

- Converts both the news headline and scraped article titles into tensor representations.
- Helps determine the degree of similarity between input and existing headlines, contributing to the fake news classification.

4.SYSTEM SPECIFICATIONS

Operating System

- Preferred: Windows 10 or above / Ubuntu 20.04 or above.
- Compatibility: Cross-platform (Windows, Linux, macOS)

Programming Language

- Python3.10+

  Python is chosen for its simplicity, extensive machine learning and web development libraries, and its community-driven ecosystem. It is particularly well-suited for rapid prototyping of AI-powered web applications.

5.LIBRARIES AND FRAME WORKS

- Flask – Used to develop the backend web server, handle HTTP routes, and manage user requests and responses.
- sentence-transformers – Provides a pre-trained Sentence-BERT model to convert sentences into semantic vector embeddings.
- requests – Sends HTTP requests to the Google Custom Search API.
- bs4 (BeautifulSoup4) – Used for web scraping, specifically to extract article titles from retrieved URLs.
- scikit-learn – Used for computing cosine similarity scores between sentence embeddings.
- json – Manages configuration files and handles structured data.
- re (regex) – Used for basic text preprocessing and cleaning

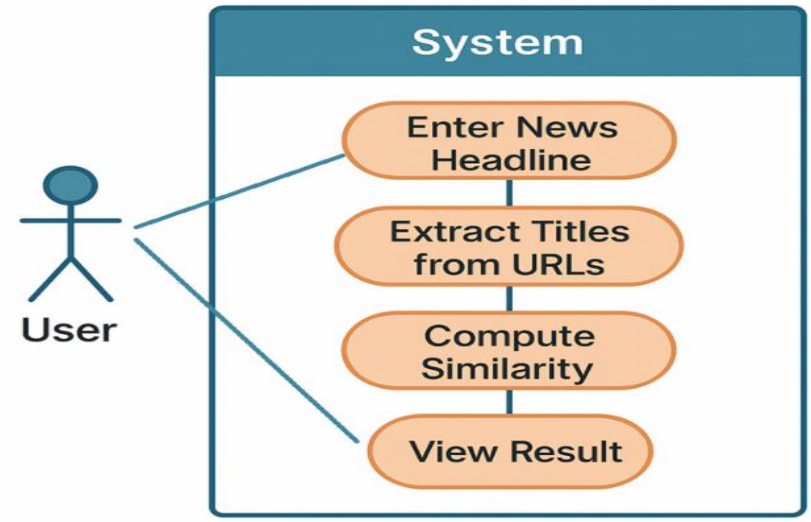
Web Technologies

- HTML5 – Used to design the user-facing input form and result pages.
- CSS3 – Styles the frontend interface for usability and accessibility.
- Jinja2 – Flask's default templating engine, used to render dynamic result pages.

Development Tools

- Code Editor: Visual Studio Code / PyCharm
- Browser: Chrome / Firefox (for testing the application)
- Package Manager: pip (Python Package Installer)
- API Console: Google Cloud Console (to manage the Custom Search API)

6. SYSTEM DESIGN



System design is the blueprint that defines the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. The design process is crucial to ensuring that the proposed solution is robust, scalable, and adaptable to evolving requirements. In the context of the Fake News Detection System, the design revolves around.

6.1 UML Diagrams

Unified Modeling Language (UML) is a visual language used in software engineering to show how a system works. It helps developers, designers, and others understand, design, and document different parts of the application. UML diagrams are like blueprints that make it easier to explain the system's structure and how it behaves.

For the Fake News Detection System, UML diagrams are used to show how the system works, how users interact with it, and how data moves between different parts. The diagrams are made in a simple and clear way, focusing on the main logic and user interactions.

6.2 Use Case Diagram

The Use Case Diagram is a type of UML diagram that shows what the system does from the user's point of view. It gives a simple overview of the main features of the system and how the user interacts with them. In the Fake News Detection System, the main actor is the user, who enters a news headline to check if it's real or fake. This diagram helps show the step-by-step process the user follows, starting from entering the headline to getting view results.

6.3 Class Diagram

The Class Diagram is a UML diagram that shows the basic structure of a system. It includes the main parts (called classes) of the program, their data (attributes), the actions they perform (methods), and how they are connected.

In the Fake News Detection System, the class diagram shows how different parts like input, search, scraping, analysis, and result handling work together to check if a news headline is real or fake

6.4 Sequence Diagram

The Sequence Diagram is a UML diagram that shows how different parts of the system interact over time. It focuses on the order of steps that happen from the start to the end of a process.

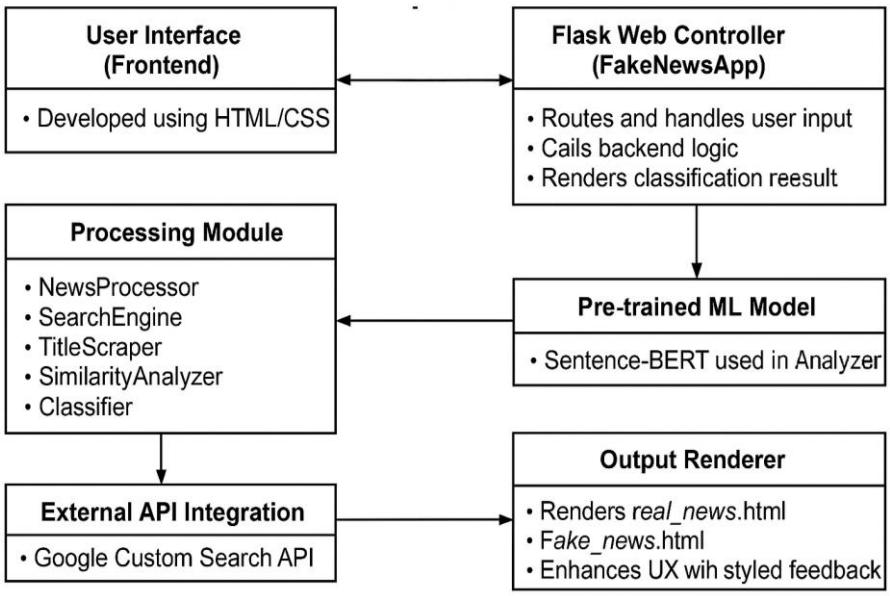
In the Fake News Detection System, this diagram shows the full process — from when the user enters a news headline to when they get a result saying it's real or fake.

6.5 Collaboration Diagram

A Collaboration Diagram (also called a Communication Diagram) is a UML diagram that shows how different parts (objects) in the system work together by sending messages to each other. It focuses on the structure of the system and how each part is connected and communicates.

The result is sent back through the Fake News App to the frontend, where it is displayed to the User.

6.6 Component Diagram



7.TEST CASES

Feature	Status
News headline input validation	Passed
URL retrieval via Google API	Passed
Title scraping and filtering	Passed
Sentence-BERT similarity model	Passed
Result classification logic	Passed
Frontend display and routing	Passed
Exception handling and fallback	Passed

8.CONCLUSIONS

This is developed for a web-based Fake News Detection System to combat digital misinformation by allowing users to input news headlines and receive real-time authenticity classifications. Leveraging Google Custom Search API for live data retrieval, BeautifulSoup for web scraping, and Sentence-BERT for semantic analysis, the system compares user input with current, publicly available information, providing accurate assessments through an intuitive interface. Its modular design and reliance on real-time data, rather than static datasets, make it adaptable and effective, with potential for future enhancements like full article verification and multilingual support, solidifying its role as a valuable tool in media literacy and ethical information technology.

9.REFERENCES

1. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
2. Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics.
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … & Polosukhin, I. (2017). Attention is All You Need. In Advances in Neural Information Processing Systems (NeurIPS).
4. 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.
5. Zhou, X., & Zafarani, R. (2019). Fake News Detection: A Survey. ACM Computing Surveys (CSUR), 53(5), 1–40.
6. Lazer, D. M. J., Baum, M. A., Grinberg, N., Friedland, L., Joseph, K., Hobbs, W., & Mattsson, C. (2018). The Science of Fake News. Science, 359(6380), 1094–1096.
7. Google Developers. (2024). Custom Search JSON API Documentation. Retrieved from: https://developers.google.com/custom-search/v1/overview
8. BeautifulSoup4 Documentation. (2024). bs4: BeautifulSoup – Python library for pulling data out of HTML and XML files. Retrieved from: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
9. Flask Documentation. (2024). Flask: Web Development One Drop at a Time. Retrieved from: https://flask.palletsprojects.com/

10. Hugging Face. (2024). sentence-transformers Library. Retrieved from: https://www.sbert.net/

11. Python Software Foundation. (2024). Official Python Documentation. Retrieved from: https://docs.python.org/

12. OpenAI. (2024). ChatGPT and GPT-based Language Models for Code Assistance and Reasoning.

13. Snopes. (2024). Fact-Checking Platform. Retrieved from: https://www.snopes.com/

14. PolitiFact. (2024). Poynter Institute's Fact-Checking Initiative. Retrieved from: https://www.politifact.com/