Verifyer

Fake news Detector

Submitted for

Statistical Machine Learning CSET211

Submitted by:

E23CSEU0130 Kaashvi Varma

Submitted to:

Mrs. Sushmita Das

July-Dec 2024

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BENNETT UNIVERSITY

THE TIMES GROUP

INDEX

Abstract **3**
Introduction **4**
Literature Survey **5**
Methodology **7**
Hardware/ Software Required **12**
Experimental Results **13**
Conclusions **15**
Future Scope **16**
Links **17**

Abstract

In the era of rapidly proliferating information, distinguishing credible news from misinformation has become increasingly critical. This project presents a Fake News Detection System leveraging machine learning techniques to assess the authenticity of news articles. It utilizes multiple machine learning models to distinguish between fake and real news. By leveraging a TF-IDF vectorizer for feature extraction and applying diverse classifiers, the system analyses textual patterns, language cues, and inconsistencies to determine authenticity. Comparative analysis of the models ensures robust performance and highlights the most effective approach for detecting misinformation. The solution is deployed using a user-friendly interface built with Streamlit, enabling seamless user interaction for inputting and verifying news content. Designed to combat the spread of misinformation, this tool provides an efficient and accessible means for individuals and organizations to verify news integrity. Future enhancements include expanding training data, incorporating advanced natural language processing techniques, and adapting the model for multilingual support to address the global challenge of fake news.

Introduction

Purpose of the report:

This report aims to:

- Highlight the challenges posed by fake news in the digital era.
- Present the design and application of multiple machine learning models for detecting fake news.
- Evaluate the performance of these models using standard metrics and identify the most effective approach.
- Serve as a reference for researchers, developers, and organizations interested in combating misinformation through automated solutions.

By addressing the growing need for reliable fake news detection, this report contributes to the ongoing efforts to mitigate the spread of misinformation and promote informed decision-making in society.

Problem Statement:

In the digital age, the widespread dissemination of information through online platforms has become a double-edged sword. While it enables rapid access to knowledge and news, it also amplifies the spread of misinformation and fake news. The propagation of fake news not only misguides individuals but also poses serious threats to society, influencing public opinion, political stability, and social harmony.

Objective:

This project aims to address the pressing issue of fake news by developing a Fake News Detection System that leverages machine learning models to differentiate between real and fake news articles. By analyzing textual patterns and linguistic features, the system provides a reliable solution for detecting misinformation. The project uses multiple classification models and a TF-IDF vectorizer for feature extraction, ensuring a robust and accurate detection process.

Significance:

The significance of this project lies in its potential to combat the misinformation epidemic and provide users with a tool to verify news authenticity. The system not only contributes to mitigating the adverse effects of fake news but also serves as a stepping stone for more advanced and scalable solutions in the field of automated fact-checking.

Literature Survey

1. A smart system for Fake News Detection Using Machine Learning

- Authors: Anjali Jain; Avinash Shakya; Harsh Khatter; Amit Kumar Gupta
- **Publisher**: IEEE
- Year: 2019

- Abstract:
  This paper demonstrates a model and the methodology for fake news detection. With the help of Machine learning and natural language processing, author tried to aggregate the news and later determine whether the news is real or fake using Support Vector Machine. The results of the proposed model is compared with existing models. The proposed model is working well and defining the correctness of results upto 93.6% of accuracy.

2. Fake News Detection Using Machine Learning Approaches

- Authors: Z Khanam, B N Alwasel, H Sirafi and M Rashid
- Publisher: IOP Publishing Ltd
- Year: 2021

- Abstract:
  This paper makes an analysis of the research related to fake news detection and explores the traditional machine learning models to choose the best, in order to create a model of a product with supervised machine learning algorithm, that can classify fake news as true or false, by using tools like python scikit-learn, NLP for textual analysis.

3. Fake News detection Using Machine Learning

- Authors: Nihel Fatima Baarir; Abdelhamid Djeffal
- Publisher: IEEE
- Year: 2021

- Abstract: In this work, the author has proposed a system for Fake news detection that uses machine learning techniques. They used term frequency-inverse document frequency (TF-IDF) of bag of words and n-grams as feature extraction technique, and Support Vector Machine (SVM) as a classifier. They have also proposed a dataset of fake and true news to train the proposed system.

4. Advancing Fake News Detection: Hybrid Deep Learning With FastText and Explainable AI

- Authors: Ehtesham Hashmi; Sule Yildirim Yayilgan; Muhammad Mudassar Yamin; Subhan Ali; Mohamed Abomhara

- Publisher: IEEE
- Year: 2024
- Abstract: This study introduces a robust approach for fake news detection utilizing three publicly available datasets: WELFake, FakeNewsNet, and FakeNewsPrediction. We integrated FastText word embeddings with various Machine Learning and Deep Learning methods, further refining these algorithms with regularization and hyperparameter optimization to mitigate overfitting and promote model generalization. Notably, a hybrid model combining Convolutional Neural Networks and Long Short-Term Memory, enriched with FastText embeddings, surpassed other techniques in classification performance across all datasets, registering accuracy and F1-scores of 0.99, 0.97, and 0.99, respectively. Additionally, we utilized state-of-the-art transformer-based models such as BERT, XLNet, and RoBERTa, enhancing them through hyperparameter adjustments. These transformer models, surpassing traditional RNN-based frameworks, excel in managing syntactic nuances, thus aiding in semantic interpretation. In the concluding phase, explainable AI modeling was employed using Local Interpretable Model-Agnostic Explanations, and Latent Dirichlet Allocation to gain deeper insights into the model's decision-making process.

5. Related articles and research papers:

a. In December 2021, Scientific Reports published a research article titled "**New explainability method for BERT-based model in fake news detection**" by Mateusz Szczepański, Marek Pawlicki, Rafał Kozik and Michał Choraś

b. A literature review titled "**The relationship of artificial intelligence (AI) with fake news detection (FND)**" was published by Emerald Insight in October 2023 by Abid Iqbal, Khurram Shahzad, Shakeel Ahmad Khan, Muhammad Shahzad Chaudhry . DOI: https://doi.org/10.1108/GKMC-07-2023-0264. ISSN: 2514-9342

c. Academic journal titled "**Model for Fake News Detection Using AI Technique**" published by International Journal of Safety & Security Engineering, 2023, Vol 13, Issue 1, p121. Authors: Rao, Kanusu Srinivasa; Challa, Ratnakumari; Sagar, B. J. Job Karuna. ISSN: 2041-9031. DOI: 10.18280/ijsse.130114

Methodology

Overview

Google Colaboratory and VS Code have been used as the development environments. The methodology for developing the Fake News Detection System involves a systematic approach, starting from data acquisition to the deployment of a machine learning-based solution. The process is divided into several key stages

1. Data Collection and Preprocessing
2. Feature Engineering
3. Model Development
4. Evaluation and Comparison
5. Deployment

Let's explore each of these aspects in greater depth

1. Data Collection and Preprocessing

The dataset used for this project was sourced from **Kaggle**. It comprises of labelled news articles, categorized as either "real" or "fake,". It includes attributes such as titles, text (article content), and their corresponding labels. This dataset provides a balanced and diverse set of examples (real: 3171 entries, fake: 3164 entries), essential for training an effective fake news detection model. The dataset is first saved on Google Drive, then imported into Google Colab using mounting.

To prepare the data for analysis, several preprocessing steps were applied:

- **Data Cleaning:** Removed unnecessary elements such as HTML tags, special characters, extra spaces, characters outside the ASCII range and numbers.
- **Tokenization:** Split the text into individual words or tokens for further processing.
- **Lowercasing:** Standardized all text to lowercase to ensure uniformity and eliminate case sensitivity.
- **Stopword Removal:** Filtered out common words (e.g., "the," "is," "and") that do not contribute significantly to the meaning.
- **Stemming/Lemmatization:** Reduced words to their root forms to handle variations (e.g., "running" → "run").
- **Sentiment Analysis:** Provides sentiment scores with **polarity** (negative to positive, -1 to 1) and **subjectivity** (objective to subjective, 0 to 1). TextBlob has been used for this purpose
- **Splitting the data:** the dataset was split into training and testing subsets to facilitate model training and evaluation. Training- 80%, Testing- 20%

2. *Feature Engineering*

Feature extraction is critical for transforming raw text into a format suitable for machine learning algorithms. For this project, the **TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer** was used:

- **Term Frequency (TF):** Measures how often a term appears in a document.
- **Inverse Document Frequency (IDF):** Assigns scores to terms depending on the frequency of their appearance across the entire corpus. Higher weight is given to terms that are less frequent, highlighting distinctive words.

It is important to get rid of stopwords when using TF-IDF vectorizer.
This approach ensured the creation of meaningful numerical representations that capture the essence of each article.

3. *Model Deployment*

The following models were trained on the processed dataset:

i. Bernoulli Naïve Bayes
ii. Logistic Regression
iii. Decision tree Classifier
iv. Random forest
v. Voting classifier (Ensemble Model)

Let's explore each one of them in detail:

i. Bernoulli Naive Bayes:

Bernoulli Naïve Bayes is a probabilistic algorithm based on Bayes' Theorem, designed for binary data. It assumes that the presence or absence of specific features (e.g., words in a document) contributes independently to the class prediction. It works well with TF-IDF vectorized data.
Bernoulli Naïve Bayes is based on Bayes' Theorem:

$$P(C|X) = \frac{P(X|C).P(C)}{P(X)}$$

Where:
P(C|X) = probability of class C given features X
P(X|C) = probability of features X occurring given class C
P(C) = probability of class C
P(X) = probability of feature X

Bernoulli Distribution:
The Bernoulli Naïve Bayes model assumes that the features follow a **Bernoulli distribution**. Each feature can take only two values:

- 1: The feature (e.g., a word) is present in the document.
- 0: The feature is absent in the document.

ii. Logistic Regression:

Logistic Regression is a linear model for binary classification that estimates the probability of a sample belonging to a class using the sigmoid function.

Linear Model:

Logistic Regression starts with a linear combination of the input features:

$z = w_1x_1+w_2x_2+\ldots\ldots\ldots\ldots+w_nx_n+b$

Where:

- x_i: Feature values.
- w_i: Model weights (coefficients).
- b: Bias term.
- z: Linear output.

Logistic (Sigmoid) Function:

To map the linear output (z) to a probability between 0 and 1, the sigmoid function is applied:

$$\sigma(z)=\frac{1}{1+e\text{^}-z}$$

This transforms the linear model output into a probability:

- $P(y=1|X) = \sigma(z)$
- $P(y=0|X) = 1-\sigma(z)$

Based on the probability from the sigmoid function, a threshold is applied (commonly 0.5) to assign a class:

- If $P(y=1|X) \geq 0.5$, predict class 1.
- If $P(y=1|X) < 0.5$, predict class 0.

iii. **Decision Tree Classifier:**

A **Decision Tree** is a supervised machine learning algorithm used for classification and regression tasks. It works by splitting the data into subsets based on feature values, forming a tree-like structure where each internal node represents a decision on a feature, and each leaf node represents the final output (class or value).

Components:

a. **Root Node**:
   - The starting point of the tree.
   - Represents the entire dataset and selects the best feature to split the data.

b. **Internal Nodes**:
   - Points where the dataset is split based on a specific condition (feature and threshold).

c. **Leaf Nodes**:
   - Terminal nodes that provide the final output (classification label or regression value).

d. **Branches**:
   - Connections between nodes that represent the outcomes of decisions.

Steps:

a. Splitting:

- The algorithm evaluates all possible splits for a feature to find the one that best separates the data into homogeneous groups.
- Common criteria for splitting include:
    - Gini Impurity: Measures the impurity of a dataset. Lower impurity means better splits.
    - Entropy: Entropy measures the disorder or uncertainty in a dataset.
    - Variance Reduction: Measures how much the variance of the target variable decreases after a split.
- The tree continues splitting recursively, creating child nodes until a stopping condition is met.

b. Prediction:

- Traverse the Tree**:** Start at the root node and evaluate the decision rules at each internal node based on the input features.
- Follow the Branches**:** Move down the tree based on the decision conditions until a leaf node is reached.
- Output**:** The class label or value associated with the leaf node is the final prediction.

iv. **Random Forest**

Random Forest is a supervised machine learning algorithm that combines multiple **decision trees** to produce a more robust and accurate prediction. It uses a process called **ensemble learning**, where the output of several models is aggregated to improve performance and reduce overfitting. Each tree independently makes a prediction, and the final output is based on the majority vote (for classification)

Random Forest uses bagging, where each tree is trained on a random subset of the training data sampled with replacement. This creates diversity among the trees, reducing overfitting. At each split in a tree, only a random subset of features is considered, adding further randomness and diversity. This ensures trees are not overly reliant on any particular feature.

Steps:

a. Training Phase:

- The Random Forest builds multiple Decision Trees using different bootstrap samples and random subsets of features.
- One tree might focus more on sentiment scores, while another might emphasize keyword presence.

b. Prediction Phase

- Each tree independently predicts a class label.
- The final prediction is based on a **majority vote** across all trees.

Final Prediction (Class)=Mode of Predictions from All Trees

v. **Voting Classifier**
It is a type of ensemble model in machine learning. It combines the predictions from multiple models (called "base estimators") to improve overall accuracy and robustness. It is not a standalone algorithm but a strategy for combining the outputs of other models.
The strategy we have used to combine models is soft voting (The final prediction is the class with the highest average probability.)

4. *Model Evaluation*

i. Metrics Used:

- Accuracy Score: Evaluated the percentage of correct predictions.
- Classification Report: Provided precision, recall, and F1-score for each class (real, fake).
- Confusion Matrix: Illustrated true positives, true negatives, false positives, and false negatives.

ii. Comparison:

- The models were compared based on their evaluation metrics to select the best-performing one.
- Emphasis was placed on achieving a balance between precision and recall for the fake news class.

5. *Deployment*

- The final step involved creating an accessible interface for users to interact with the system. Using **Streamlit**, a web application was developed that allows users to input a news article's text and receive real-time predictions about its authenticity.
- The application provides a clear and simple interface, ensuring usability for a non-technical audience.
- It integrates the selected machine learning model, seamlessly handling data preprocessing, prediction, and result display behind the scenes.

This deployment transforms the machine learning solution into a practical tool for combating fake news, offering both accuracy and convenience to users.

Hardware/ Software Required

1. Hardware

i. Processor (CPU)
ii. RAM
iii. Storage
iv. Internet Connectivity

2. Software

i. Operating System: Windows 10
ii. Programming Languages: Python 3.12.6
iii. IDE:
   - VS Code: for front end
   - Google Colab: for data preprocessing and model training
iv. Libraries:
   a) Machine Learning Libraries:
      - Scikit- Learn: for building and evaluating models
      - Numpy: for numerical computations
      - Pandas: for data manipulation and analysis
   b) Natural Language Processing (NLP) Libraries:
      - TextBlob: for sentiment analysis and preprocessing
      - NLTK: for tokenisation and stopword removal
      - re: For pattern matching, text cleaning, and regex-based preprocessing
   c) Feature Extraction Tools:
      - TfidfVectorizer from Scikit-learn
   d) Visualisation Library:
      - Matplotlib
      - Seaborn
      - Plotly
v. Model Deployment framework: Streamlit, for creating an interactive web app
vi. Dependency Manager: pip for managing Python packages
vii. Version control: Git and GitHub for version control and collaboration
viii. Web Browser: Microsoft Edge for accessing the web app
ix. ChatGPT was utilized in the development of this project to assist with drafting, refining, and structuring the report, as well as providing insights and clarifications on key concepts

Experimental Results

Performance Metrics

we have evaluated the performance of the models using the following metrics:

- **Accuracy**: Percentage of correctly classified instances.
- **Precision**: The proportion of true positive predictions (e.g., fake news classified as fake).
- **Recall**: The ability to correctly identify fake news.
- **F1-Score**: The harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix**:

Results for Each Model

Here are the results obtained from the evaluation:

Model	Accuracy	Precision		Recall		F1 Score	
Bernoulli NB	0.71	Real	0.73	Real	0.69	Real	0.71
		Fake	0.70	Fake	0.75	Fake	0.72
Logistic Regression	0.83	Real	0.83	Real	0.86	Real	0.84
		Fake	0.85	Fake	0.82	Fake	0.84
Decision Tree	0.77	Real	0.78	Real	0.78	Real	0.78
		Fake	0.77	Fake	0.78	Fake	0.78
Random Forest	0.86	Real	0.87	Real	0.87	Real	0.87
		Fake	0.87	Fake	0.87	Fake	0.87
Ensemble Model	0.85	Real	0.86	Real	0.85	Real	0.85
		Fake	0.85	Fake	0.86	Fake	0.85

Confusion Matrix:

	Predicted Real	Predicted Fake
Actual Real	TN	FP
Actual Fake	FN	TP

- True Positives (TP): fake news correctly classified as fake.
- True Negatives (TN): real news correctly classified as real.
- False Positives (FP): real news incorrectly classified as fake.
- False Negatives (FN): fake news incorrectly classified as real.

Confusion Matrix of the models:

1. Bernoulli NB: [[468 160] [197 442]]
2. Logistic Regression: [[516, 112] [91,548]]
3. Decision Tree: [[485,143] [139,500]]
4. Random Forest: [[545,83] [85,554]]
5. Ensemble Model: [[540,88] [99,540]]

Conclusions

This project aimed to develop a robust system for detecting fake news by leveraging various machine learning models and natural language processing techniques. The following conclusions can be drawn from the experiments conducted and the results obtained:

1. ***Effectiveness of Machine Learning Models***
   - Among the evaluated models, the Random Forest Classifier demonstrated the highest accuracy and balanced performance across all evaluation metrics, making it the most suitable choice for deployment in real-world scenarios.
   - Logistic Regression and Bernoulli Naive Bayes also performed well, offering simpler and faster alternatives with slightly lower accuracy.
   - The Decision Tree Classifier, while interpretable, showed the lowest accuracy, likely due to its tendency to overfit.

2. ***Significance of Preprocessing and Feature Engineering***
   - Text preprocessing techniques like tokenization, stopword removal, and TF-IDF vectorization were essential for transforming raw text into meaningful features, significantly improving model performance.
   - The use of regular expressions (re library) for cleaning data further ensured the quality of the input text, which is critical in handling noisy datasets like news articles.

3. ***Evaluation and Comparisons***
   - The evaluation metrics, including accuracy, precision, recall, and F1-score, provided a comprehensive understanding of model performance. Confusion matrices highlighted areas where misclassifications occurred, offering insights into potential areas for improvement.

4. ***Practical Implications***
   - This system can serve as a foundation for tools used by journalists, researchers, and the public to identify fake news and mitigate its spread.
   - The interactive front-end developed using Streamlit ensures user accessibility and ease of use, allowing for seamless real-time predictions.

5. ***Challenges and Limitations***
   - While the system performs well on the dataset, its real-world applicability depends on the quality and diversity of training data. Future improvements can focus on incorporating larger and more diverse datasets.
   - Models may require periodic retraining to adapt to the evolving nature of fake news patterns and language.

<u>Future Scope</u>

To further enhance the system's performance and utility, future work could focus on:

1. ***Incorporating Advanced Models:***
   - Exploring deep learning models like Recurrent Neural Networks (RNNs), LSTMs, or transformer-based models (e.g., BERT) for improved text understanding.

2. ***Expanding Dataset:***
   - Using a larger and more diverse dataset, including news from different languages and regions, to improve generalizability.

3. ***Real-Time Deployment:***
   - Optimizing the model for real-time fake news detection in production environments, including API integration and scalability.

4. ***Explainability:***
   - Adding mechanisms to explain the model's predictions, which can increase user trust and understanding of the system.

Links

GitHub Repository:

https://github.com/kaashvivarma/Verifyer.git

Link to Deployed project:

verifyer-at4i8skbxgsxxpnhwlqes3.streamlit.app

Thank you!