BCS: Veritas Vergil Report

Naina Bhalla
Roll Number: 240674

1 Overview

I have developed a complete pipeline for detecting fake news using a combination of rule-based processing and machine learning. A custom tokenizer, part-of-speech (POS) tagger, and lemmatizer were built from scratch to handle informal language, repeated characters, and word variations commonly found in fake news.

Text features were extracted using both Bag-of-Words and TF-IDF methods, and two models—Naive Bayes and Linear SVM—were trained for classification. Results showed that the SVM model with TF-IDF features performed best, achieving high accuracy and reliably separating real from fake news.

2 Process

2.1 Custom Tokenizer Design

The custom tokenizer was developed to robustly process informal, sensational, and noisy news text. Its design incorporated the following rules:

- **Case Normalization:** All text was converted to lowercase to ensure uniformity and reduce sparsity in the vocabulary.
- **Contraction Handling:** Common English contractions (e.g. "can't", "won't") were expanded to their full forms ("can not", "will not") using a predefined mapping. The list was obtained from Wikipedia (LINK) . Some contractions were broken such as "could t". They were rejoined and expanded too after tokenization.
- **Punctuation and Emoticon Preservation:** Punctuation marks and emoticons were retained as separate tokens, as they are important in conveying sentiment or emphasis, especially in deceptive news.
- **Tokenization by Regex:** Regular expressions were used to split text into tokens, capturing words, numbers, punctuation, and emoticons as distinct elements.
- **Repeated-Character Normalization:** Elongated words (e.g., "sooooo") were detected using regex and normalized by collapsing the repeated sequence into the base character plus a special token indicating the repetition count (e.g., "so <REPEAT:5>"). This highlights emotional exaggeration, often present in fake news.

2.2 Rule-Based POS Tagger (Mini Tagger)

A lightweight, rule-based part-of-speech (POS) tagger was implemented, relying on word suffixes.

- Some common verbs without any specific rule were set aside such as is,are, etc.
- Words ending in -ing, -ed : Verbs
- Words ending in -ly,-ous,-ful,-ive,-able,-ic : Adjective
- Words ending in -ness,-ment,-ion : Nouns
- The REPEAT: tokens : Meta
- For tokens not yet tagged, iterate through a list of suffix patterns (e.g., "-ing", "-ed", "-ly") and assign the associated POS tag if a match is found.
- If the token starts with an uppercase letter and is still untagged: Proper Noun
- If the token starts with a lowercase letter and is still untagged: Noun
- If no rule applies: Other

2.3 Custom Stemmer/Lemmatizer

A rule-based stemmer/lemmatizer was designed to reduce words to their base forms, using the POS tags as guides to prevent over stemming and retain meaning in words.

Verb Lemmatization:

1. Removed "-ing", "-es" and "-ed" suffixes to retrieve the base verb (e.g., "running" → "run").
2. Removing -ing and -es only if word is longer than 4 letters to prevent sing from becoming s and similar cases

Noun Lemmatization:

1. Replace "-ies" endings with "y" (e.g., "stories" → "story").
2. Remove "-es" from words ending in "-es".
3. Remove a final "-s" from words that are not "ss" and are longer than three letters (e.g., "cats" → "cat" but "boss" remains "boss").

Adjective/Adverb Lemmatization:

1. Remove "-est" from words ending in "-est" (if the word is longer than five letters).
2. Remove "-er" from words ending in "-er" (if the word is longer than four letters).

e.g. biggest" → "big"

2.4 Feature Extraction

Two main feature extraction methods were implemented:

- **Bag-of-Words (BoW):**

  Each document was represented as a vector of word counts, capturing the frequency of each token. This just counts the raw frequency.

- **TF-IDF (Term Frequency-Inverse Document Frequency):**

  This method weighed tokens by their frequency in a document and their rarity across the corpus, highlighting distinctive terms. This helps in deciding the emphasis of words.

2.5 Classification

Two classifiers were trained to distinguish real from fake news:

2.5.1 Naive Bayes (Multinomial):

This model assumed conditional independence between features, making it particularly well-suited for high-dimensional text data. It uses the frequency of words to estimate the likelihood of news being real or fake. Despite its simplicity, it performed well.

2.5.2 Linear Support Vector Machine (SVM):

Unlike Naive Bayes, which reasons probabilistically, SVM directly focuses on boundary separation. Its strength becomes evident in scenarios where elongated words or abnormal token patterns distorted the natural language distribution.

2.5.3 Performance Metrics:

Accuracy, precision, recall, and F1-score were calculated to evaluate and compare classifier performance. Along with that, confusion matrix and ROC curves were plotted.

2.6 Visualization and Analysis

- **Token Frequency Distributions:**

  Visualized the most common tokens in each class to identify distinguishing vocabulary. I first removed the common english stop-words using NLTK to only visualize the important words rather than focusing on stop-words such as articles.

- **Word Clouds:**

  Created for both real and fake news to reveal prominent terms.

- **REPEAT:n Token Distribution:**

  Examined the prevalence of repeated-character tokens, often indicative of sensationalism.

- **Confusion Matrix and ROC Curves:**

  Used to assess and compare classifier effectiveness.

2.7 Impact of Custom Rules

- **Repeated-Character Normalization:**

  Helped highlight and reduce the influence of exaggerated language, improving classifier precision.

- **POS-Guided Lemmatization:**

  Enhanced generalization by grouping inflected forms, leading to improved recall.

2.8 Comparative Analysis

Four models were evaluated using two feature extraction methods: Bag-of-Words (BoW) and TF-IDF. The models tested were Naive Bayes and Support Vector Machine (SVM). Performance was measured using accuracy, precision, recall, F1-score, and ROC AUC.

Model	Accuracy	Precision	Recall	F1-score	ROC AUC
Naive Bayes (BoW)	0.946	0.956	0.945	0.951	0.975
Naive Bayes (TF-IDF)	0.936	0.925	0.962	0.943	0.982
SVM (BoW)	0.994	0.994	0.995	0.994	0.998
SVM (TF-IDF)	0.993	0.993	0.995	0.994	0.999

2.8.1 SVM Superiority: Both SVM models (using BoW and TF-IDF) significantly outperformed Naive Bayes across all metrics, achieving near-perfect accuracy (above 0.99), F1-score, and ROC AUC. This indicates SVM's strong ability to distinguish real from fake news in the dataset.

2.8.2 Feature Extraction Effects: For Naive Bayes, BoW yielded slightly higher precision and accuracy, while TF-IDF improved recall and ROC AUC, indicating it better captures subtle differences in word importance. For SVM, both feature sets performed exceptionally well, with TF-IDF offering a marginally higher ROC AUC.

2.8.3 High Overall Performance: All models demonstrated robust performance, with Naive Bayes exceeding 0.93 accuracy and SVM models exceeding 0.99.

2.8.4 Interpretation: The SVM models, especially when paired with TF-IDF features, are highly effective for this fake news detection task. The consistently high scores also suggest that the custom preprocessing pipeline—including tokenization, normalization, and lemmatization—contributes meaningfully to model performance.

2.8.5 Summary: The comparative analysis demonstrates that SVM with TF-IDF features is the optimal approach for this dataset, achieving the highest accuracy and discrimination.