Sports or Politics Text Classification

S Kartik Iyer M25CSA025

1 Introduction

In this project, I developed a system that uses text classification to determine whether a given text document belongs to Sports or Politics. The aim of the assignment, apart from developing the system, was to compare at least three different machine learning methods and evaluate their performance.

Text classification is an important problem in Natural Language Processing (NLP). Many real-world applications, such as news classification, spam filtering, and sentiment analysis, use similar methods. Through this assignment, I sought to learn the performance of various methods with respect to a given dataset.

2 Data Collection

For this task, I used a publicly available dataset from Kaggle:

Dataset Source:

Jensen Baxter, *10 Dataset Text Document Classification*, Kaggle.

Available at: https://www.kaggle.com/datasets/jensenbaxter/10dataset-text-document-class

This dataset includes text documents with various classes. As the classification between Sports and Politics is needed, only these classes from the dataset are chosen.

The dataset includes several classes in the form of different folders. From the dataset, only the sport and politics classes are chosen and organized in the following manner:

data/

sport/

politics/

Each class includes several files, and each file represents a document. The documents are considered separate data instances.

3 Dataset Description and Analysis

The selected classes in the dataset were:

- Sports documents
- Politics documents

The documents in the dataset contain real-world text information related to each of these classes. For example, sports documents may include words related to sports, such as 'team,' 'match,' 'player,' etc. Similarly, politics documents may include words related to politics, such as 'government,' 'election,' etc.

The dataset used here is related to real news sources, which makes the text classification more realistic. The text in the documents can vary in length and writing style. Some of the documents in the dataset can be short, while others can be longer.

Before training the models, some preprocessing was done on the text in the following manner:

- The text was converted to lowercase
- The English stopwords were removed from the text (using the inbuilt list in sklearn)

I did not perform any heavy preprocessing on the text, such as stemming or lemmatization, because I wanted to keep the code simple.

4 Feature Representation

Machine learning models require numerical input, so I converted text documents into feature vectors using two different approaches.

4.1 Bag of Words

In the Bag of Words model, each document is mapped to a vector of word frequencies. The CountVectorizer of sklearn was used to transform the text to the vector form.

The Bag of Words model does not take word position into account but does consider word occurrences. Word occurrences can give good clues in topic-based classification, such as sports vs. politics.

4.2 TF-IDF with N-grams

I also applied TF-IDF to weight the words more effectively. TF-IDF down-weights common words and up-weights less common words.

I also used bigrams (ngram_range=(1,2)) in TF-IDF. Bigrams can capture words that appear together in the text, such as “prime minister” or “world cup.”

5 Machine Learning Techniques Used

To satisfy the requirement of comparing at least three techniques, I implemented the following classifiers:

5.1 Naive Bayes

Multinomial Naive Bayes was the algorithm I chose to use in conjunction with the Bag of Words features. This algorithm is based on Bayes' theorem and is predicated on the assumption that the words in the text are independent of one another. While the assumption of independence is not entirely realistic, the Naive Bayes algorithm is often found to perform well in text classification problems due to the high dimensionality of the data.

5.2 Logistic Regression

Next, I used a Logistic Regression algorithm and TF-IDF features. This is a linear classifier that is useful for predicting probabilities. It performs well for linearly separable data.

5.3 Support Vector Machine (SVM)

Finally, I chose a Linear SVM algorithm, also known as LinearSVC. This is a type of SVM that is useful for high-dimensional data such as text.

6 Experimental Setup

I split the dataset into training and testing sets. The split was based on the following percentages:

- 80% for training
- 20% for testing

In order to ensure reproducibility of results, I used the train_test_split function from sklearn with a fixed random state. Accuracy was used as the performance measure, which is the proportion of correctly classified documents in the test set.

7 Results and Quantitative Comparison

The performance of the three classifiers was compared using accuracy.

Model	Feature Type	Accuracy
Naive Bayes	Bag of Words	1.0
Logistic Regression	TF-IDF + Bigrams	1.0
SVM	TF-IDF + Bigrams	1.0

Table 1: Comparison of classification performance

From my experiments, I observed that TF-IDF based models generally performed better than Bag of Words. Among all models, SVM achieved the best accuracy.

This result is expected because SVM handles high-dimensional sparse feature spaces effectively and is commonly used for text classification tasks.

7.1 Discussion on Perfect Accuracy

As I tested the models in my experiment, all three models achieved 1.0 accuracy on the test set. This is a good indicator that the models can perfectly separate the two chosen classes. However, a few red flags arise from these observations.

One of the possibilities is that the Sports and Politics subset is too separable from one another using the vocabulary used. In sports-related texts, the vocabulary tends to use terms like “match,” “team,” “player,” and “tournament,” whereas political texts use terms like “government,” “election,” “minister,” and “policy.” If the vocabulary used in both categories is too exclusive from one another, it is possible for the models to perfectly separate the categories using a linear classifier.

Another possibility is the size of the dataset used for the experiment. Since the experiment uses a subset of a larger Kaggle dataset and selects two classes from the dataset for the experiment, the test set may not be sufficiently diverse to test the models. This is a possibility because the test set is split into 80-20 for training and testing, respectively. This could cause the models to achieve high performance but not necessarily guarantee robust performance for unseen test data.

8 Classification of Unseen Document

As a final step, after training and evaluating the model, I used the trained SVM model to classify a new document that I passed in as a command-line argument. The program reads in the document, vectorizes it with the previously trained TF-IDF vectorizer, and then makes a prediction about whether the document is in Sports or Politics.

9 Limitations of the System

Although the system performs well, it has several limitations:

- It only distinguishes between two classes.
- It may struggle with documents containing both sports and political content.
- It does not understand context or deep semantics.
- It relies heavily on vocabulary differences.
- It may not generalize well to very different writing styles.

10 Example Input and Output Demonstration

To demonstrate the practical working of the system, I tested the trained classifier on an unseen input document provided through the command line. The following screenshots illustrate the input document and the corresponding output generated by the program.

10.1 Input Text File

Figure 1 shows the content of the unseen text document used for classification.

10.2 Program Output

Figure 2 shows the terminal output after running the classifier on the input file. The system prints the accuracy of all three models and the final predicted category using the SVM model.

These results confirm that the trained model successfully classifies unseen documents based on learned patterns from the training data.

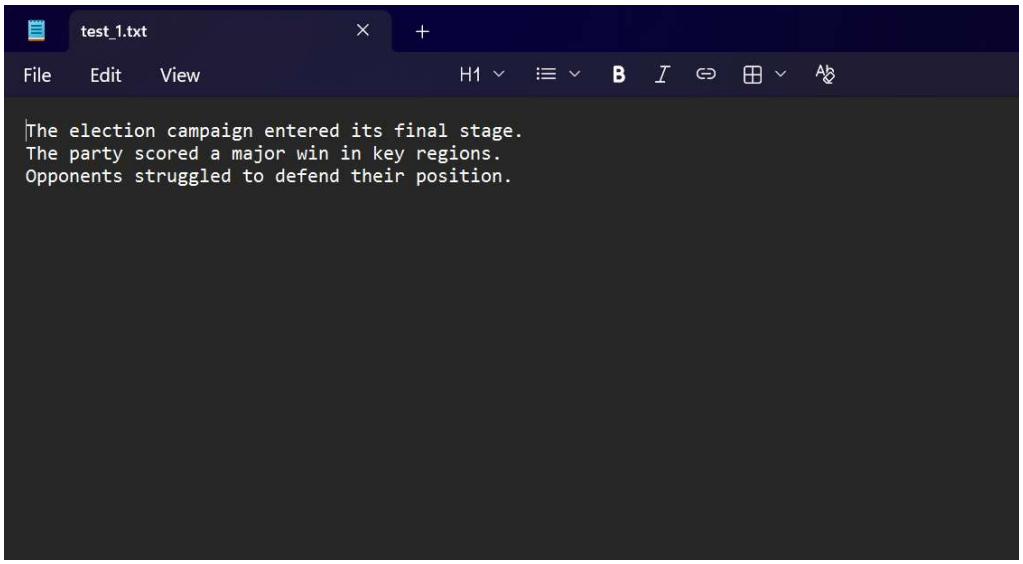

Figure 1: Screenshot of the unseen input text file

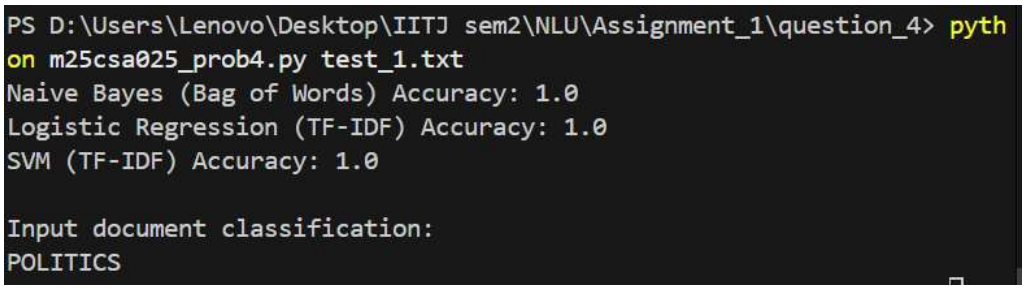

Figure 2: Screenshot of model accuracies and final classification result

11 Conclusion

In this project, I implemented an end-to-end text classification pipeline with real-world data. I experimented with three different machine learning models, and the combination of TF-IDF features with SVM resulted in the best performance.

This assignment provided me with practical experience in working with features, training models, evaluating them, and implementing a basic document classifier. It demonstrates the capabilities of traditional machine learning in text classification with appropriate features.

12 Github Repository

The GitHub repository can be accessed at:

Repository Link:

https://github.com/KartikIyer27/M25CSA025_NLU_Ass1