Text Mining and Natural Language Processing
Group Home Assignment

By Tinu Jijo & Marco Malchiodi

Date: 21/04/2025

Curriculum: ITLB365_25S

Institute: IBS – International Business School

GitHub repository: https://github.com/MarcoMalchiodi/NLP-Home-Assignment

INDEX

1) Introduction

Modern media outlets tend to be characterized by a wide spectrum of political leanings, something which is often a source of controversial topics. Whether conservative, progressing or non-leaning in orientation, these biases are often manifested through subtle linguistic patterns, choice of vocabulary and less pronounced nuances. The purpose of this project is to explore the availability of Natural Language Processing (NLP) models and methodologies in order to analyse news articles from distinguished news outlets and thereby assess whether computational models can indeed detect such biases, thus contributing to the development of potential tools with the purpose of assessing more reliable sources.

The structure of this process follows the standard NLP pipeline. Text data is obtained from various news websites via web-scraping practices, and then labelled based on the political leaning of their respective sources. The raw texts then undergo a process of data cleaning and preprocessing, making them suitable for the following word embedding phase. These techniques are essential in order to turn text data into numerical data that can be processed by Machine Learning (ML) models. The final stage entails the fine-tuning and application of the models themselves, leading selective comparisons, and finally implementing simple tf-idf, neural embedding and hybrid search engines.

2) Data & Data Preprocessing

Two Python scripts have been built and adopted with data scraping purposes: *url_scraping* and *news_scraping*. The former used a simple request-based API service (newsapi.org) to collect several news articles from a series of pre-arranged URLs, labelled these articles based on their website of origin (i.e. western_conservative, western_progressive, non_western), and finally loaded the collected information into a json file ; the latter, on the other side, imported essential libraries for web-scraping purposes, such as Selenium, and then accessed each URL from the previously created json files in order to convert the HTML page content into raw text data. A total of 1164 articles have been collected for this project.

An additional Python script, *cleaner.py*, has been created with the purpose of eliminating redundant blank spaces left by the Selenium library. The final data cleaning and preprocessing steps have been taken within this project's main file: *news_classifier.ipynb*

The first step taken involves the transformation of all unique categories into numerical values via One-Hot-Encoding. This process enables the conversion of categorical variables into a numerical format. This is necessary as it is the only format that ML models are capable of processing. A binary column categorization is applied to each instance, where the feature matching with a given category receives the value of '1', otherwise becoming '0'.

Before proceeding to the word-embedding phase for the text data, a further data cleaning session is introduced, as certain unique symbols or characters may have not been removed by the previously imported methods. This step was refined via the Natural Language Toolkit (NLTK) library, which has rendered the removal of stop-words possible.

Finally, a word-embedding technique has been applied to the cleaned text data. Word embedding is a method to transform words into number with an approach that maintains words similar to one another in close proximity. These vectors assist computers to understand meanings, relationships, and patterns between words by analysing their occurrences within large text datasets. The word-embedding method imported is Word2Vec.

3) Model Fitting and Evaluation

This project employs four models on the dataset and the best performing model will be chosen for achieving the best result. The models are:

1. Naïve Bayes

Naïve Bayes is one of the widely used algorithms for text classification problem which is grounded on Bayes theorem. This algorithm classify data by using conditional probability where it assumes each feature are independent of each other. As this project is a text classification problem Multinomial Naive Bayes type will be used. This is the variant where it classifies the problem based on the frequency of words in the dataset.

2. Support Vector Machines (SVM)

Support Vector Machine is a popular and efficient supervised machine learning algorithm used for both the regression and classification problems. SVMs models are widely applied as it often outperforms the Naïve Bayes algorithm. It classifies text by finding the best hyperplane which separate the text into classes. SVMs are also famous for its ability to capture the non-linear relationships between the texts.

3. Random Forest

Random Forest is the third algorithm employed in this project for classifying text. This algorithm uses the ensemble decision trees for both classification and regression tasks. The final class is predicted followed by the voting system by each decision tree for classification problems. Efficiency in handling high dimensional data and finding non-linear relationships are making this algorithm more reliable.

4. Recurrent Neural Networks (RNNs)

Deep learning models such as Recurrent Neural Network and Convolutional Neural Network can also be leveraged for the purpose of text classification. Long Short-Term Memory a variant of Recurrent Neural Networks is also applied in this project other than the machine learning models. LSTM is capable of solving the limitations of traditional RNNs models and machine learning models by capturing long distance relationships between texts and need for rigorous feature engineering.

3.5) Result and Analysis

1. Naïve Bayes

Naïve Bayes model has achieved a score of 92.6 percentage on test dataset, which is very good in predicting the class accurately. Upon getting the predicted probabilities for the article 'former president donald trump seeking sweeping...', the model predicted the class western_progressive with 1.0 of probability. The model shows a high confidence in this class when comparing to other labels such as western_conservative and non_western with a probability of 0.0. This means the model was able to capture strong features in this article for classifying it as western_progressive.

2. Support Vector Machines (SVM)

The SVM algorithm shows a higher accuracy of 98.2 percentage on the test dataset. This also indicates the ability of model to predict the class of articles very precisely. Furthermore, when analysing the predicted probabilities for the article 'former president donald trump seeking sweeping...', the result shows a high confidence with a probability of .995 for the class western_progressive and a lower probability for the classes western_conservative and non_western. This means the model is able to find strong evidence for predicting it as the class western_progressive. High accuracy percentage and good predicted probability makes this model to achieve a good result.

3. Random Forest

Analysing the results from Random Forest algorithm also reveals a high accuracy score. Accuracy score for random forest model is 97.8 percentage which is applied on test dataset. This indicates the ability of this algorithm to classify news articles with a high accuracy. Upon checking the predicted probabilities for the news article 'former president donald trump seeking sweeping...', the model predicted a high probability for the class label western_progressive with a confidence score of .99. Confidence score for non_western and western_conservative received .01 and 0 respectively. This result shows the ability of model to capture relevant features.

4. Recurrent Neural Networks (RNNs)

The model employed with Recurrent Neural Network also gained a good accuracy on test data. Test accuracy for the RNN model is 78 percentage, which assure a good result in classifying data. When analysing training loss from first to final epoch, it decreased from 1.02 to 0.15 which is also a good factor in this model. The training accuracy of this model achieved a good score of 95 percentage. Upon analysing the predicted probabilities for the new article 'former president donald trump seeking sweeping...', the model predicted class label as western_progressive with a high confidence score of .99 and other labels received a lower probability.

Model Comparison

Model	Test Accuracy	Sample Prediction	Predicted Probability
1. Naïve Bayes	95%	western_progressive	1.0
2. SVM	98%	western_progressive	.99
3. Random Forest	97%	western_progressive	.99
4. RNN	78%	western_progressive	.99

Support Vector Machine model is the best performing model, when analysing the results. SVM achieved the highest test accuracy of 98 percentage among all applied algorithms. All models predicted the class label as western_progressive for the sample news article with higher probability. The ability of support vector machine model to capture the non-linear dependencies definitely helped in achieving this high-test accuracy.

4) Search Engine Implementation

The final section represents an implementation of search engine methodology tailored specifically for political news articles, accompanied by an evaluation system which adopts Large Language Model (LLM) judgments. This approach implements and compares three distinct search methods: a traditional TF-IDF approach, a neural embedding technique, and lastly a hybrid model that combines both approaches. The remaining part of this technical report analyses each component of the implementation, discuss its architectural decisions, evaluate its advantages and limitations, and consider its potential applications in the realm of political sentiment analysis.

The system is made up of five main components which work together to provide insightful search features for political content. At its foundation lies a dataset of news articles, which has been pre-processed and categorized based on its political leanings. The framework then includes three distinct search engines, each employing different information retrieval techniques. Finally, an evaluation framework powered by LLM judgments provides a directive to assess the quality and relevance of search results.

1. TF-IDF Engine

The TFIDEngine class adopts a classic information retrieval system using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. This method represents documents as points in a high-dimensional space, where each dimension reflects how important a word is in the document compared to how often it appears in the whole collection.

The implementation follows the following steps: 1) initialization fits the vectorizer to the entire document collection, creating a term-document matrix; and 2) search operations convert queries into the same vector space and compute cosine similarities to find the closest documents.

The advantages of this approach include its computational efficiency for both indexing and querying, interpretability of results, and proven effectiveness for keyword-based searches. For political news retrieval, this method would perform well when queries contain distinctive terminology used particularly by certain political perspectives (e.g., "illegal aliens" vs. "undocumented immigrants").

However, it is important to point out that such method entails some limitations that may constitute a set-back for the purpose of this project. Semantic relationships between various terms cannot be captured (e.g., understanding that "Obamacare" and "Affordable Care Act" refer to the same policy) and it is sensitive to vocabulary differences that may not reflect actual content differences. Moreover, it struggles with conceptual queries that don't map directly to specific terminology.

2. Neural Embedding Engine (Modern Semantic Search)

The NeuralEmbeddingSE class represents a more recent approach which adopts vector embeddings generated by a pre-trained sentence transformer model ('all-MiniLM-L6-v2'). This methodology captures semantic relationships between words and documents by representing them in a continuous vector space where similar meanings are geometrically close.

This approach may be more effective as far as political sentiment analysis is concerned, as understanding allows matching beyond literal keyword overlap. It can identify conceptually similar content even when different terminology is used, thus better handling paraphrases, synonyms, and related concepts. Overall, it is more effective at concept understanding.

The 'all-MiniLM-L6-v2' was chosen for being a distilled version of larger models that balances performance with computational efficiency, making it practical for our purposes. For political analysis, this means being able to recognize that articles discussing "tax relief" and "tax cuts for the wealthy" might represent different political perspectives on similar policies.

That being said, here, too, some challenges can be encountered. For instance, the most relevant aspect would be the higher computational requirement encountered during document processing. Performance itself is heavily reliant on the pre-training of the underlying model. Moreover, this method may sometimes retrieve semantically related but politically irrelevant content.

3. Hybrid Engine (Combined Approach)

The HybridSE class is an implementation which combines results from both the TF-IDF and neural approaches through a weighted average. This hybrid methodology attempts to capture the strengths of both approaches while significantly reducing their individual weaknesses.

The implementation strategy behind the hybrid search engine begins by retrieving twice as many results from both the TF-IDF and neural embedding subsystems, which guarantees there are enough candidate documents to choose from. The system then adopts an adjustable parameter (i.e. *alpha*) to control the balance between the two methods. This allows flexibility in determining how much influence each approach should have on the final results. The scores from both methods are then combined using a simple weighted average.

This hybrid approach is especially effective for retrieving political news articles, as the neural embedding component understands the overall meaning of the text, which allows it to recognize different ways of expressing similar ideas. By combining these two advantages, the system is able to find documents that are not only relevant in terms of specific terminology but also in terms of their broader meaning and context.

4. Evaluation Framework

By leveraging LLM capabilities, the system can evaluate result relevance in a more nuanced way than traditional metrics like precision and recall. The evaluation system includes basic error handling, defaulting to a neutral rating when something goes wrong, and supports both individual and group-level evaluations. What makes this approach particularly reliable is its ability to pick up on political significance beyond just surface-level topics, thus granting it a more human-like understanding.

That being said, here, too, we encounter some limitations: the system relies on the language model's political understanding, which may not always be reliable, and API failures automatically return neutral scores, potentially resulting in skewed results.

5. Results

The sample search results have yielded highly differing results. On average, the most promising scores have been achieved by the Neural Embedding Engine, with results shifting within the range of 0.370-0.430 depending on the query proposed, thus pointing to a significantly stronger semantic recognition when compared to the other two methods, although not quite satisfactory. The worst performer was TF-IDF engine, with scores rarely above the 0.200 threshold. Nonetheless, it should once again be pointed out the results may have been heavily influenced by a dataset which may have been unreliable for the aims and purposes of this final section.

5) Conclusion

Political sentiment analysis is a significantly delicate field of natural language processing, as the interpretation of the data itself can be highly subjective and prone to personal biases. The implementation of the ML models in the third section yielded considerably positive results, with accuracy scores rarely falling below the 0.85 threshold. The model with the highest performance was SVM with an average score of 0.98, whereas the worst performance was obtained with the application of an RNN model with an average score of 0.78. Ultimately, the dataset contained news sources which may fall under the category of “mainstream media”, therefore, in order to further test the models’ predictive performance, it would be advisable to build new datasets containing a wider variety of news sources. Finally, less satisfactory were the results obtained by the search engine implementations. Even the Neural Embedding Engine, which yielded the best scores out the three engines proposed, failed nonetheless to surpass the 0.5 score threshold, thus showcasing a poor affinity between the data collected and the parameters adopted in the final evaluation.