Media Bias

CMPE 255

Professor -

December 13, 2024

Nikhil Khanchandani, Chint Patel, Dustin Nguyen, Vikranth Jakamukala

Abstract:

In today's world, media bias has become an issue, causing the public perception to be skewed. News articles and media content are often crafted with specific language to subtly favor one viewpoint more than the other, making it hard for readers to discern unbiased information. The choice of words, sentence structure, and framing in these articles can mislead readers, encouraging them to adopt particular views without a fair or balanced perspective. This bias not only distorts the truth but also diminishes readers' ability to form independent, well-informed opinions.

Introduction:

Our solution aims to address this issue by implementing a sentiment analysis tool that analyzes the language and tone used in various news articles. By evaluating the sentiment embedded in the wording, this tool will provide insights into how different topics are presented across sources. This sentiment analysis model will assess articles' positivity, negativity, and neutrality, highlighting any leanings that may sway readers. By providing users with an unbiased overview of media sentiment, our solution empowers readers to make more informed choices, ultimately fostering a more transparent media landscape.

Related Work:

Sentimental analysis is a very interesting subject. There are some related works that are similar to our project. Starting with TextBlob which is the most relevant one to our project. It’s best used for simple sentiment analysis. It provides features like polarity scores, and is easy to

use for text preprocessing. There's also Amazon Comprehend, which is mainly used for enterprise software level projects. It scales the Sentiment analysis for such projects. It has features like identifying the overall sentiment of the article and it is compatible with many languages. There are also complex related works like Google Natural Language API, and IBM Watson NLU.

Data:

Our project used a sentiment analysis dataset from Kaggle to train and test our bias detection model. The dataset included text data with sentences or short articles with a given sentiment such as positive, negative, or neutral. The dataset was the Sentiment Analysis Dataset by Abhishek Shrivastava which is used for building sentiment classification models.

The data includes thousands of sentences or paragraphs extracted from sources such as news articles, reviews, etc. It contains over ten thousand rows of text with the structure Text, Sentiment Label which consists of the actual sentence and then the label indicating it to be positive, negative, or neutral. To prepare the dataset for model training, we took steps to clean the data. We removed Stopwords which are common words like "the," "and," "is,", etc. using NLTK to reduce the noise in our model. Lowercasing all text and removing special characters such as punctuation, numbers, etc. The sentences then were split into words for analysis. The preprocessing improved the performance of TF-IDF feature extraction so it only focuses on words that have actual meaning. Additionally, the dataset was split 80% into training, and 20% into testing to train our logistic regression model efficiently and this made sure our distribution was balanced to prevent bias.

Methods:

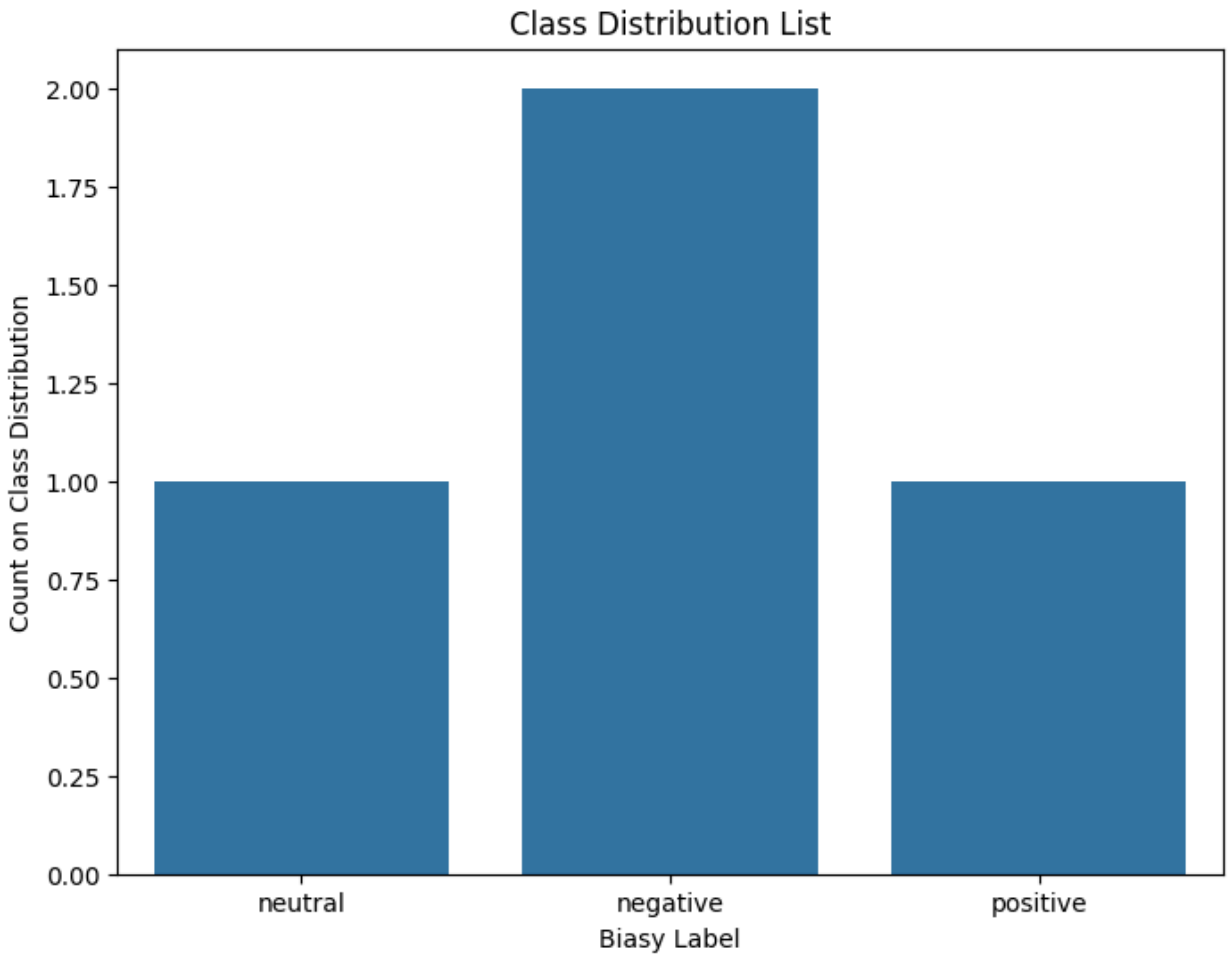
To tackle this daunting task of media bias detection, we created a sentiment analysis tool that evaluates the tone and language of the article to provide a clear understanding of it. To do this we trained a machine learning model to classify articles as positive, negative, or neutral. Users can freely input the URL of their article to determine the bias of the piece dynamically.

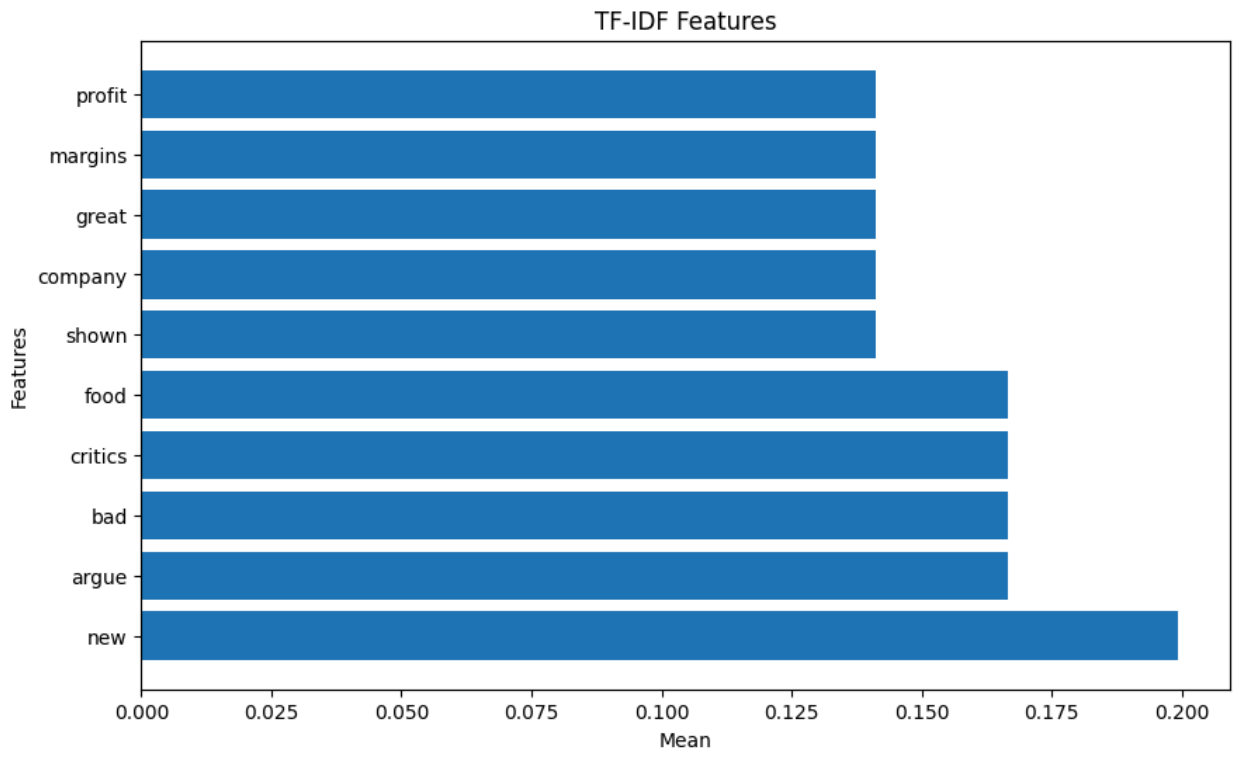
We utilize machine learning, natural language processing, and data preprocessing techniques to find bias in the content. The steps we employed include Data preparation, Feature Extraction, MOdeling, Evaluation, and Bias Analysis via URLs. Data preparation: we remove noise like special characters, stopwords, fillers, and extra spaces. This is then used to visualize the class distribution of different words and find some level of bias there. Feature extraction is done through TF-IDF vectorization. This is where articles are vectorized with Term Frequency-Inverse Document Frequency models. This finds the importance of a word while diminishing the more common and less meaningful words. For the Modeling we used Logistic Regression where the model is trained on preprocessed text to classify sentiment. The dataset is split into both testing and training sets to achieve this result. Evaluation metrics like precision, recall, and F1-score were used to determine the performance of our model. We included a confusion matrix to distinguish between the three bias categories. Finally, a scraping function is used to extract articles from URLs that the user can provide. The extracted text is then preprocessed and fed to the model for classification.

The rationale behind these approaches is to ensure noise-free inputs with data preprocessing. TF-IDF is used to find the important words for better bias detection. Finally, Logistic Regression is used to balance the simplicity of performance and good results. Although there were some alternatives we could have used. A pre-trained model like BERT could capture

the subtle nuances in the text. This would come at a higher cost in computation. Other custom architectures like LSTM or CNN. Again this would consume a lot of resources and require a more extensive labeled dataset.

Overall the solution we used balances robustness with simplicity. The preprocessing of the articles, trains machine learning models, and integrates bias detection through web scraping. For future advancement, we could adopt more advanced models and get better datasets to improve the contextual understanding of the model.





Experiments and Results:

We compared the logistic regression-based models with two methods which are Rule-Based and Naive Bayes Classifier. With the Rule-Based method, we are using a set of keywords that were predefined to classify the text which makes it simpler to make a reasonable baseline. The Naive Bayes Classifier is a probabilistic classifier used to train on the TF-IDF feature, which is a standard choice for classification tasks mainly used for text and this provides a meaningful benchmark. This project model was examined using the metrics like accuracy, precision, recall and the F1-score. With the result that we have gotten, we see that we are outperformed our baseline with the score of accuracy of 82.5%, 62.5% for rule-based, and 70% for Naive Bayes. This was shown by the approaches we had using the TF-IDF representation and logistic ability to capture the feature importance for bias classification.

To review the importance of the components in our system, we had a study done called ablation study by removing noise that would be in the data such as:

- No stopword removal: Stopwords such as “I”, “is,” “the,” and “an” were kept in the preprocessing step.
- Without TF-IDF Feature: employed the bank full of words to represent
- Reduced Feature set: TF-IDF vectorizer was reduced to 2500 features

With each test, we found that by not removing the stopwords, we have reduced the accuracy by 4.5%, without the TF-IDF feature giving a 7% drop in accuracy and reducing the feature degrading the importance of the classification.

With the enhancement of the model’s performance, we want to experiment the different parameters for the logistic regression by checking the regularization strength we want to test to find the optimal balance between the bias and variance. The other method we used was solver selection to check the performance using different solvers. The setup had an accuracy of 82.5% and a balanced precision and recall.

What we used to visualize the models were the Confusion Matrix Heatmap and the Feature Importance. With the heatmap we wanted to highlight the numbers of each message like true positive, true negative, false positive, and false negative to help identify the patterns of misclassification. For the Feature Importance, we use influential words for each of the classes and exact them based on the highest TF-iDF scores then create a bar chart to represent the top 10 TF-IDF features. This performed well, but sometimes runs into errors where the language is nuanced or context-dependent.

To analyze the failure modes, we look at

- Ambiguous Language: Articles using neutral language that hints at bias through subtle phrasing that causes the misclassification
- Out-of-Vocabulary Words: TF-IDF vectorizer need more training for phrases that were not in the training data
- Short articles: Text that had too little words to compare

These observations can benefit the model by understanding the context more.

In conclusion we demonstrated that our approaches effectively can solve the problem with text bias classification, outperforms the baseline, and offers good results, but we want to improve this further by:

- Data augmentation: Expand the training dataset to include more language patterns and nuances to improve the generalization
- Contextual Embeddings: Explore the model by examining the combination of TF-IDF features with contextual embeddings like BERT
- Error Mitigation: Address the failure modes with the targeted data collection

By going through our examination, we aim to refine the accuracy and apply this in the real-world application.

Conclusion:

For our project, we have successfully developed a sentimental analysis tool that measures the tone and bias towards an article. Our tool provides a solution for the users where they can analyze the tone which could be positive, negative, or neutral over many web sources. Our goal was to have users evaluate information of what the article tone is which makes the user's

understanding for the article easier. Our approach hopes to help promote accountability in media reports which could influence the public perception. In the future, we will expand our tool to use more dynamic and complex sentiment models which can help users to engage and understand the news much better.