Addressing Political Bias in News Articles with Multinomial Regression

Artificial Intelligence CS4100

Taylor Stevens and Anjali Tanna

Fall 2023

GitHub Repository

Contents

Purpose

In today's digital landscape, the pervasive presence of biased content within online articles has become an acute societal concern[4]. The proliferation of this biased information across various online platforms has granted narrow narratives the power to shape perceptions, beliefs, and decisions[1]. Recognizing this, the mission of this project is not merely to acknowledge the existence of bias but to help combat its influence through an innovative approach. The endeavor seeks to address this issue by crafting a solution that acts quickly in the identification of bias[5]. Understanding that mitigating bias demands a multifaceted strategy anchored in data-driven methodologies, the proposed solution hinges on the creation and utilization of a meticulously curated dataset improved with labeled information on diverse articles, revealing their inherent biases. The dataset used in this project serves as the foundation for this initiative. Central to the deployed strategy is the development and deployment of a robust multinomial regression algorithm. This algorithmic framework in the context of this project is designed to predict and classify the bias categories of unseen articles with speed. Its predictive capabilities, powered by annotated data, can help in taking a step in the direction of mitigating bias within the online sphere[2]. By successfully classifying articles with speed and accuracy, the broader vision of the project is to foster a more informed, objective, and equitable digital discourse. Through collaborative efforts, the project aspires to cultivate a healthier information ecosystem that pushes past present limitations and offers more credibility to unbiased sources while simultaneously offering equity in information for readers.

Problem Statement and Use Cases

Problem Statement

The objective of the project centers on the optimization of a multinomial regression model (MLRM) tailored for predicting bias in online articles. It is hypothesized that although political bias is a complex problem, a MLRM will be able to classify a satisfiable amount of articles that it is presented with. This model operates on a dataset consisting of crucial columns: *topic*, *source*, *bias_score*, and *paragraph_vectors* (reflecting article content), each numerically encoded and normalized. The optimization would include elevating the model's efficacy and ensuring dependable predictions when confronted with unseen articles. In general, the multinomial regression model is known for its ability to predict probabilities of group membership across multiple classes or categories. In the case of this project, it is tuned to recognize and predict the bias category of online articles, drawing insights from the features inherent in the dataset columns. Across multiple iterations, the model provides probability estimates to articles such that they are in various bias categories, with the goal of reducing error as it trains itself. The three categories that this project's model is training to predict includes both *left*, *center*, and *right* labels, translated into the numerical scores of 0, 1, and 2 respectively. Each score showcases the dominant bias ingrained within the article based on the provided features. Aside from the goal of having a high accuracy rate in classifying articles, it is important to understand the model's output and the sources of misclassification.

Use Cases

The three use cases outline below showcase the model's applicability and significance:

1. Media Oversight and Fact-Checking: It can be leveraged to aid in identifying and rectifying biased content for maintaining journalistic integrity[3].
2. Educational Material Curation: The model can be employed to curate unbiased or balanced learning resources, fostering critical thinking and balanced perspectives among students.
3. Social Media Moderation: It can be used to help mitigate the dissemination of biased or misleading information[3].

This model, with increased accuracy and interpretability, represents a tool across various domains that promises a more equitable and transparent assessment of bias within online content.

System Architecture and Requirements

Database Selection and Cleaning

The dataset utilized for training and testing the model is sourced from a publicly available repository on Google Datasets, accessible through the Hugging Face platform[7]. Composed of 13 comprehensive columns, including important attributes such as *topic*, *source*, *bias*, *url*, *title*, *date*, *authors*, *content*, *source_url*, and *bias_text*, this dataset stands as a foundation for the analysis to be undertaken post training. The project's success heavily relied upon pre-labeled data, which this repository was able to provide[7]. Without the availability of labeled data, the model would have been performing unsupervised learning, which would have introduced ambiguity and complexity, potentially undermining the analysis. The supervised learning approach that was taken ensured a more straightforward trajectory in discerning bias classifications, particularly in the political spectrum. Furthermore, the existence of labeled data allowed for rigorous testing, validating the accuracy and efficacy of the classification model. The dataset's depth lies not only in its pre-labeling but also in the diversity of information encapsulated within its columns. Being able to leverage the diversity of attributes as features during model training enabled a more nuanced and comprehensive assessment of biases from the model. Such nuance was imperative for training a MLRM as generally these types of models are not the strongest available to classify complex ideas. The dataset's bias distribution represents a broad spectrum of the ways in which bias can permeate articles. Understanding the dataset's distributions proved instrumental in understanding the model's sensitivity to different biases. Additionally, examining bias splits by source shed light on the inherent biases affiliated with specific platforms or publishers, crucial insights that guided the understanding of how biases manifest across diverse sources. The following is a summary of the bias splits within the total dataset and the breakdown of biases categorized by their respective sources, as shown in diagrams 1 and 2. The Total Dataset Bias Splits gives a total of 31% left biased articles, 23% center bias articles, and 46% right bias articles. From the top 5 sources, the sources can be categorized as left for ABC News, center for BBC, right for CBN, left for CNN, and center for Reuters.

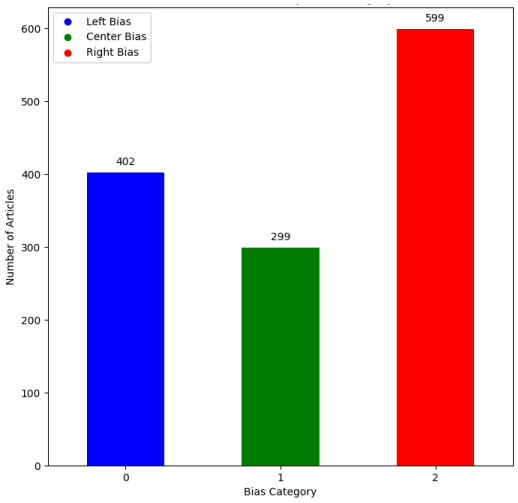

Diagram 1 - Number of Articles by Bias Category

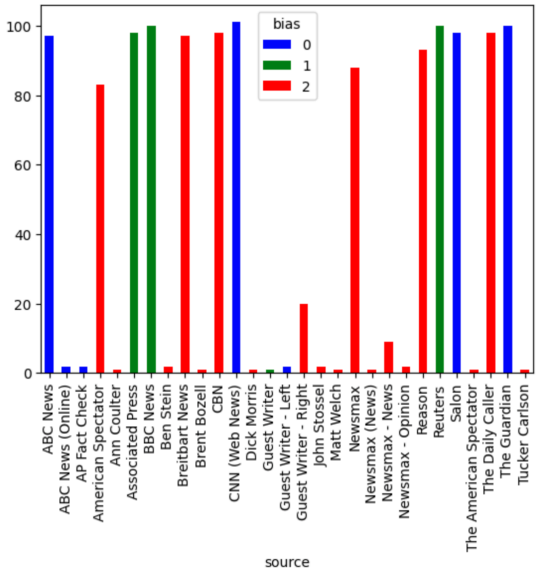

Diagram 2 - Bias Type by Source

Upon acquiring the dataset, a series of preprocessing measures were undertaken to ensure data coherence and relevance to the classification objectives. The primary aim was to refine the dataset, making it more comprehensible for the model while eliminating extraneous or potentially misleading attributes. Initially, columns prone to introducing noise or misleading the model's learning process, such as *author*, *date*, *time*, etc. were removed. This curation process aided in reducing the dataset down to attributes appropriate to the classification goals. As a result, *topic*, *source*, *bias*, *title*, and *content*, stood as the remaining columns in the dataset for the model to train with. After column reduction was complete, *title* and *content* were transformed by Doc2Vec, a powerful vectorization library. The utilization of Doc2Vec allowed for the mapping of textual data to meaningful numerical representations. The two columns were merged before undergoing the transformation by Doc2Vec, which empowered the model to capture not just word importance but the semantic meaning of the entire article including its title. Further, from this merged column, prior to applying the library, filler words, commonly known as stop words, were removed from the content. This extraction process aimed to reveal the essence of the articles, eliminating linguistic clutter that might obscure the model's understanding. This nuanced representation resulted in the creation of an output array comprising 100 feature values per data sample. The utilization of Doc2Vec, which pivoted away from simple word importance, enabled the model to grasp the intricate nuances and contextual depth embedded within each article. This was important as bias cannot simply be captured by which words an author may have chosen. After the vectorization of the text was completed, the *source* and *topic* columns underwent separate preprocessing. Each column was given its own manual vectorization, replacing word values with numerical values through mapping. Finally, the 3 vectorized columns (*source*, *topic*, and the merged *content*/*title*) underwent normalization, ensuring uniformity and consistency in scale across the feature space, a crucial step preceding model training. Prior to introducing this step, the model had significant trouble with classifying the articles, likely due to the different scales between *source*/*topic* and *content*/*title*. This sequence of preprocessing steps, ranging from column curation to semantic representation and normalization, resulted in a refined dataset optimized for the model's comprehension. By leveraging these techniques to consolidate information, the preprocessing allowed for the model training to result in significantly more accurate bias classification than before.

Design Methods

To validate fundamental assumptions and adapt the approach for handling different types of data, existing code derived from Programming Assignment 4 was modified for use by the MLRM. Unlike the original code that primarily dealt with pixel values, the adaptations made with this project allowed the model to effectively work with new types of data relevant to the project goals of classifying bias.

This adaptation process involved several key steps including creating a TrainBiasDataset Class to

accommodate the unique characteristics of the data. This dataset is structured to handle the *topic*, *source*, *bias_score*, and *paragraph_vectors* features, as opposed to the pixel structure of the original datasets in PA4. This adjustment ensures that the model could effectively learn from and make predictions on text based information. The main changes are seen within the initialization function as below, where the data is taken out of the CSV and split into test and training sets:

```
class TrainBiasDataset(Dataset):
  def __init__(self, data_path, classes):
      data = pd.read_csv(data_path)
      classes = range(classes)
      split_data = []
      self.feature_size = 0
      for ind in data.index:
            p_vectors = data.iloc[ind]
                  [[str(i) for i in
                  range(100)]].to_numpy()
            X = data.iloc[ind][['topic',
                  'source']].to_numpy()
            X = np.concatenate(
                  (np.array(X),
                  np.array(p_vectors)))
            if ind == 0:
                  self.feature_size =
                  len(X)
            y = data.iloc[ind]
                  ['bias_score']
            split_data += [(X, y)]

      self.xs, self.ys = zip(*split_data)

      X_train, X_test, y_train, y_test =
            train_test_split(self.xs,
            self.ys, test_size=0.2,
            random_state=42)

      self.xs = X_train
      self.ys = y_train
```

The Multinomial Regression Model, originally designed for a different context, was updated to handle the TrainBiasDataset, which was made to handle non-pixel data. These adjustments enabled the bias MLRM to effectively process, learn from, and make predictions based on the unique characteristics of new types of data. Utilizing the original model resulted in the following code:

```
# test both 2 and 3 class bias split
for n in [2, 3]:
      data_path =
            "normalizedDataWithCenter.csv"
            if n == 3 else
            "normalizedDataNoCenter.csv"
      train_data =
            TrainBiasDataset(data_path=
            data_path, classes=n)
      feature_size =
            train_data.feature_size
      train_model =
            MultiLogisticRegressionModel(
            num_features=feature_size,
            num_classes=n)
      accuracy_sample_frequency = 250
      sample_size = 5000
      train_model.train(train_data,
            sample_size,
            accuracy_sample_frequency)
      accuracies =
            train_model.
            get_training_accuracies()
      train_data.plot_accuracy_curve(
            eval_iters=range(0,
            sample_size,
            accuracy_sample_frequency),
            accuracies=accuracies,
            title=
            'Training Accuracy Curve')
      train_data.plot_confusion_matrix(
            train_model)
```

Results and Analysis

Results

Two Class Classification

The accuracy of 2 class (left and right bias) classification gradually increased over the training period as follows: Iteration 250, Accuracy: 0.585 → Iteration 500, Accuracy: 0.808 → Iteration 750, Accuracy: 0.826 → Iteration 1000, Accuracy: 0.829 →Iteration 2000, Accuracy: 0.835 → Iteration 3000, Accuracy: 0.838 → Iteration 4000, Accuracy: 0.836 → Iteration 5000, Accuracy: 0.846 → Final Accuracy: 0.851% (681 out of 800). This can be visualized with the figure below (Diagram 3) and the confusion matrix that follows (Diagram 4), which gives insight to the models main production challenges.

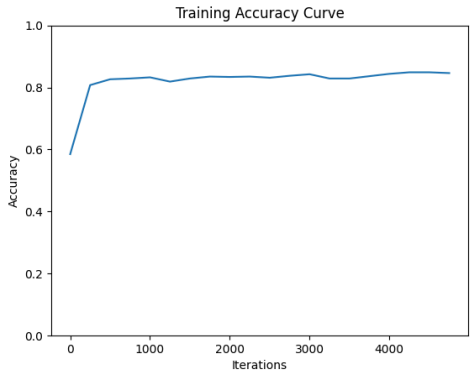

Diagram 3 - Two Class Training Accuracy Curve

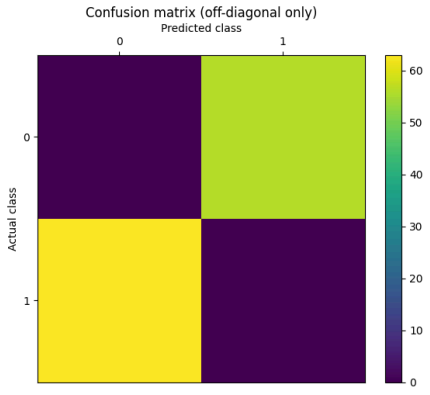

Diagram 4 - Two Class Confusion Matrix Plot

Three Class Classification

The accuracy of 3 class (left, center, and right bias) classification gradually increased over the training period as follows: Iteration 250, Accuracy: 0.234 → Iteration 500, Accuracy: 0.716 → Iteration 750, Accuracy: 0.761 → Iteration 1000, Accuracy: 0.784 → Iteration 2000, Accuracy: 0.797 → Iteration 3000, Accuracy: 0.813 → Iteration 4000, Accuracy: 0.820 → Iteration 5000, Accuracy: 0.821 → Final Accuracy: 0.819% (852 out of 1040). This can be visualized with the figure below (Diagram 5) and the confusion matrix that follows (Diagram 6), which gives insight to the models main production challenges.

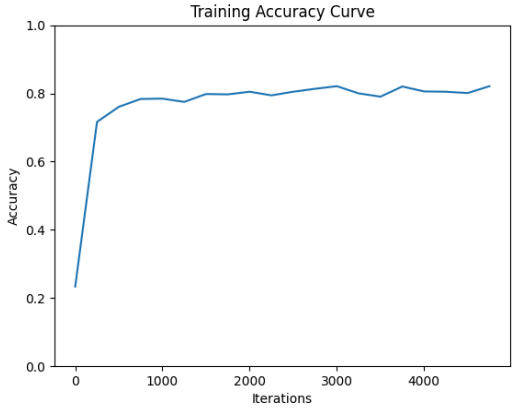

Diagram 5 - Three Class Training Accuracy Curve

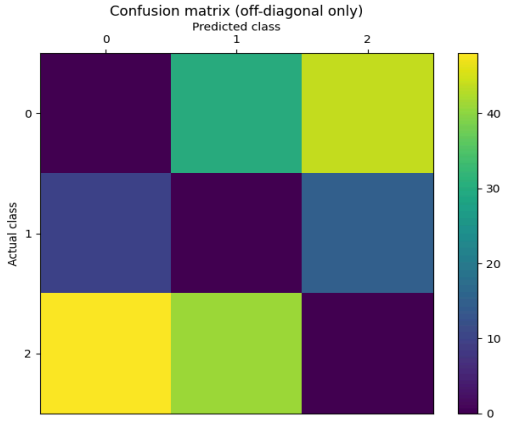

Diagram 6 - Three Class Confusion Matrix Plot

Analysis & Discussion

The exploration of a multilogistic regression model (MLRM) surfaced considerable insights into the intricate nature of identifying political bias within written articles. Despite considerable progress made through a significant learning curve, it became evident that MLRM might possess inherent limitations in capturing the nuanced intricacies of this classification process, a barrier not predicted with the hypothesis. Although there was much more success after a significant learning curve, there are notable missing pieces in the project. One of the key limitations stemmed from MLRM's assumption of linear decision boundaries between classes. Political bias, however, can manifest as a multifaceted spectrum rather than adhering to distinctly separable linear boundaries. This complexity likely resulted in the model creating oversimplified classifications that failed to encapsulate the nuanced positions and ranges of political bias present in articles.

Furthermore, MLRMs operate under the assumption of feature independence, a premise that might not hold true for political bias, which was not considered prior to analysis. This lack of feature independence could stem from not only language nuances and historical references but also contextual intricacies and rhetorical devices. Attempting to encapsulate these multifaceted aspects within a linear model framework would therefore pose significant challenges. Because MLRMs might struggle to contextualize these elements effectively, the training could result in a loss of crucial information essential for comprehending political leanings within articles. Considering these factors, while MLRM served as a valuable baseline model, achieving higher accuracies than found in this project would likely require more sophisticated approaches. Some techniques rooted in natural language processing (NLP), including deep learning architectures or ensemble methods, could likely better handle the complexity and non-linearity present in the data. Further NLP methods would likely find more success in capturing the subtle contextual cues vital for accurately identifying and classifying political bias within articles. Despite the model's inherent limitations, noteworthy insights emerged from the training. Notably, the selection of features such as source and topics significantly influenced the model's success, resulting in almost 85% accuracy in the classification of test data with just a little over 1000 samples in the dataset. Additionally, the utilization of the Doc2Vec library played a pivotal role in the model's ability to achieve satisfactory classification accuracy. The transformation of text based data into semantic representations through this library greatly contributed to the model's efficacy compared to other vectorization techniques that were tested. Without the DocToVec vectors as features, the model failed to even hit 50% classification abilities.

Alongside the difficulties with MLRM and complex classification, several challenges and unmet expectations hindered the project's progression. The scarcity of data that precisely matched the desired criteria, specifically, the unavailability of a dataset with five categories for classification, imposed limitations on the model's complexity and accuracy. It is likely however, that although more complex data could not be found, that the MLRM would have struggled to classify a 5 category political bias, as the model's accuracy witnessed a slight decline from only 2 to 3 categories. Further, the scale of the dataset, though larger than manually curated alternatives, remained insufficient for practical applications. The limited volume of around 1,300 rows potentially impacted the model's accuracy due to inadequate instances for comprehensive learning. The search for a larger dataset proved difficult, consuming significant time without yielding the desired scale of data. When attempting to supplement the small dataset, it was quickly clear that it would not be reasonable to continue in the timeframe provided as it was extremely difficult to get raw text of articles due to paywalls and other cross origin resource policies (CORP) that blocked the use of libraries such as BeautifulSoup.

In future iterations, it would be reasonable to retry this implementation with a dataset of larger scale, however it may continue to prove difficult to find data that matches the project's requirements. Additionally, the initial strategy of employing TfidfVectorizer to determine word importance and create features did not yield the expected results. When researching different NLP word vectorization techniques, the TfidfVectorizer library was encountered. It was believed that TF-IDF (Term Frequency Inverse Document Frequency) would be useful in converting the article content data into numerical data. While ample time was spent learning how to use this tool, the model's accuracy remained inadequate, not breaching 50%, and the technique proved computationally slow compared to the more successful Doc2Vec method, which only encompassed 100 features for each data point. The model evaluated with TfidfVectorizer held over 33,000 features after data processing, as it was a combination of word importance over all the words in the dataset between articles. This realization led to a substantial reevaluation of data preprocessing steps and significantly extended the preprocessing phase. While TfidfVectorizer did create a significant number of features, the accuracy remained much too low to consider this even a partial success. The accuracy of 3 class (left, center, and right bias) classification gradually increased over the training period with TfidfVectorizer as follows: Iteration 200, Accuracy: 0.23 → Iteration 400, Accuracy: 0.462 → Iteration 600, Accuracy: 0.461 → Iteration 800, Accuracy: 0.46 → Iteration 1000, Accuracy: 0.309 → Final Accuracy: 0.46% (598 out of 1300). The resulting confusion matrix for the three class classification reported as:

0 [0, 2, 400]
1 [0, 1, 298]
2 [0, 2, 597]

After failing with TfidfVectorizer, the NLP word embedding technique, Word2Vec, was the next feature extraction technique that was implemented. Word2Vec is a widely used word embedding technique and is a neural network that learns to predict the probability of a word given its context[8]. After spending valuable time converting the article content data using Word2Vec, it proved to be similar to TfidfVectorizer, and therefore not beneficial to the model's classification efforts. This library however, resulted in the discovery of Doc2Vec, which brought forth the realization that political bias might not merely be linked to word repetitions within articles. Rather, the identification of bias appears to be intricately linked to word relations, context, and broader linguistic nuances. This observation underscores the crucial role of context and word relationships in bias identification, deviating from a simplistic association with individual word occurrences. This is demonstrated between both a successful and failed identification by the model on a left and right leaning article as follows:

> "In the fallout over President Barack Obama blaming the intelligence community for the rise of the Islamic State, a new report has surfaced showing that he attended less than half of his daily intel briefings… 'It's pretty well-known that the president hasn't taken in-person intelligence briefings with any regularity since the early days of 2009,' the staffer said. 'He gets them in writing.' Unless someone very senior has been shredding the president's daily briefings and telling him that the dog ate them, highly accurate predictions about (ISIS) have been showing up in the Oval Office since before the election."

The above article was misclassified as a left leaning article when it should be correctly classified as a right leaning article. As seen in the snippet above, there are no mentions to specific right wing policies and it can be assumed that the article is classified as right leaning because of the author's shown distaste with Obama's in house practices and attention to detail around military briefings. Because there are no

concepts mentioned about how military operations should be handled or how policy should help inform military endeavors, it is clear this is one of the types of nuances that cannot be summarized so easily by an MLRM. This can be seen in contrast to a left leaning article that was properly classified by the MLRM below:

> "Republican Sen. Jeff Flake told CNN he is willing to reverse his opposition to expanding background checks for guns ... Flake said the only reason he voted no was because of his concern that the requirement for background checks on internet sales is too costly and inconvenient … Manchin and gun control advocates need to convince five senators to go from 'no' to 'yes'…Some Republicans opposed the measure out of fear that expanding background checks would put the country on a path to a national gun registry, but Flake said that is not his concern…"

Within this snippet, the author mentions a positive stance on gun control and regulations as well as background checks, a concept that is strongly and solely linked to left wing policies. This makes identification a much more clear cut option for the model, whereas both left and right leaning articles might have positive or negative views on any given candidate at different times. Just looking at these two short examples, it is clear that bias is extremely nuanced and cannot be simply divided into simple categories by just vectorized article semantics.

Overall, In future iterations, efforts should focus on securing a larger and more diverse dataset to facilitate comprehensive model training. Exploring more advanced NLP techniques capable of capturing nuanced contextual cues and relationships within articles could significantly enhance the accuracy and robustness of bias identification models. Moreover, meticulous consideration of feature selection and preprocessing methods remains integral to achieving higher accuracies in classifying political bias within written content.

Next Steps

Dataset

As touched on before, future efforts would be focused on creating a larger and more diverse dataset. This would allow for comprehensive model training as well as exploring more advanced techniques. A potential idea for creating a larger dataset would be to web scrape the allsides.com website. Allsides.com strives to expose people to information and ideas from all sides of the political spectrum so they can better understand the world and each other[6]. This would give us a larger and more diverse dataset to use on the model. Another potential idea to achieve this, would be to utilize an API in order to easily obtain the large amounts of diverse data on allsides.com.

User Interface

A second possibility to expand upon this project would be to create a user-friendly interface for the general public to use to find the bias score of a given article. Using HTML and CSS, a website could be created in conjunction with the multinomial regression model to input any article and output a bias score. The possibilities for this website would be endless, as automated comparisons between news articles and their bias scores, comparisons between different news outlets, and a variety of other statistics to exemplify the biases found within articles could be created.

Summary

To summarize, the purpose of this project was to address the concern of biased content in online articles by developing a machine learning model. The hypothesis assumed that using a multinomial regression model could effectively classify a bias score (*left, center,* or *right)* within articles found online. The algorithm relied on a diverse, meticulously created dataset with the MLRM operating on features such as *topic, source, bias_score,* and *paragraph_vectors.* Using techniques such as Doc2Vec to vectorize article content data, the MLRM was able to classify bias scores within articles with 85.1% accuracy for two-class (*left* and *right*) classification and 81.9% accuracy for three-class (*left, center,* or *right*) classification. Insightful analyses include highlighting the significance of feature selection, as accuracy skyrocketed when including the topic and source in the data. Challenges included limitations on specific NLP techniques related to word vectorization. In the future, acquiring a larger and more diverse dataset while exploring more advanced NLP techniques is suggested. Further, creating a user-friendly interface for the bias MLRM would expand the scope of its impact. Overall, this project provides valuable insights to both the bias identification within articles as well as multinomial regression models and the challenges that come with it.

Advice for Future Students

To the future students of CS4100, the biggest piece of advice would be to give yourself plenty of ideas and options when deciding the avenue for this project. There is no clear-cut path of the milestones within this project; many things are subject to change. When this project was initially introduced, our first idea was to implement a search algorithm within a map of the city of Boston to address the issue of traffic flow. We started to do research on this idea and looked for datasets, we realized that this idea may not be able to come to fruition. There were fairly limited datasets and public access available to traffic data online, so we soon realized we needed to pivot our thought process. Additionally, we agreed that we could implement an AI model that would be more extensive, and more valuable to our learning than a search algorithm. This led us to our next potential idea: addressing political bias in news articles with multinomial regression. We believed that this would give for a better opportunity to explore different tools and facets within machine learning and artificial intelligence, one that would broaden our knowledge beyond the scope of what our search algorithm idea could. Even after deciding upon article bias and multinomial regression, we were faced with issues within creating our dataset. We had to find a way to vectorize the article content data. Our first approach was to use TfidfVectorizer. This ultimately failed, as our model failed to even hit 50% classification abilities. Our next approach to vectorize the article content data was Word2Vec. This approach also did not perform as well as we thought it would. After discouragement soon approached us, we tried using Doc2Vec, which produced significantly higher results, ending up with almost 85% accuracy in the classification of test data with just a little over 1000 samples in the dataset. Given these obstacles, we urge you not to get discouraged too early in the process, and allow for adaptation as needed. It may be more work to scrap your original idea and start with a brand new one, but the learning process is more valuable than if you succeed on your first shot. Overall, take in the roadblocks with open arms and you will reach the destination you are looking for. Good luck!

References

1. DellaVigna, Stefano, and Ethan Kaplan. "The Fox News Effect: Media Bias and Voting." The Quarterly Journal of Economics 122, no. 3 (2007): 1187–1234. https://doi.org/10.1162/qjec.122.3.1187.
2. Rodrigo-Ginés, Francisco-Javier, Jorge Carrillo-de-Albornoz, and Laura Plaza. "A Systematic Review on Media Bias Detection: What Is Media Bias, How It Is Expressed, and How to Detect It." Expert Systems with Applications 237 (2024): 121641–. https://doi.org/10.1016/j.eswa.2023.121641.
3. Kartikey Pant, Tanvi Dadu, and Radhika Mamidi. 2020. Towards Detection of Subjective Bias using Contextualized Word Embeddings. In Companion Proceedings of the Web Conference 2020 (WWW '20 Companion), April 20–24, 2020, Taipei, Taiwan. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3366424.3382704
4. Cremisini, A., Aguilar, D., Finlayson, M.A. (2019). A Challenging Dataset for Bias Detection: The Case of the Crisis in the Ukraine. In: Thomson, R., Bisgin, H., Dancy, C., Hyder, A. (eds) Social, Cultural, and Behavioral Modeling. SBP-BRiMS 2019. Lecture Notes in Computer Science(), vol 11549. Springer, Cham. https://doi.org/10.1007/978-3-030-21741-9_18
5. Hamborg, Felix, Karsten Donnay, and Bela Gipp. "Automated Identification of Media Bias in News Articles: An Interdisciplinary Literature Review." International Journal on Digital Libraries 20, no. 4 (2019): 391–415. https://doi.org/10.1007/s00799-018-0261-y.
6. All About AllSides. (2023). AllSides. https://www.allsides.com/about
7. Ziems, Caleb. "CJZIEMS/Article-Bias-Prediction · Datasets at Hugging Face." cjziems/Article-Bias-Prediction Datasets at Hugging Face. Accessed December 13, 2023. https://huggingface.co/datasets/cjziems/Article-Bias-Prediction
8. Khare, Pratyush. 2023. "Deep Learning for NLP: Word2Vec, Doc2Vec, and Top2Vec Demystified." Medium, April 25, 2023. https://medium.com/mlearning-ai/deep-learning-for-nlp-word2vec-doc2vec-and-top2vec-demystified-3842b4fad5c9.

Project Repository

The project including all code referenced in this document and the databases mentioned can be found at the link https://github.com/anjali-tanna/cs4100_final_project on GitHub.