Classifying Political Bias in News Articles

Rhys Mackenzie*, Gabriel Iturriaga†, Courtney Pick‡, and Ben Kelly§
*Dalhousie University Halifax, NS, Canada
Email: rhys.mackenzie@dal.ca
†Dalhousie University, Halifax, NS, Canada
Email: gabriel.iturriaga@dal.ca
‡Dalhousie University Halifax, NS, Canada
Email: cr693886@dal.ca
§Dalhousie University Halifax, NS, Canada
Email: ben.kelly@dal.ca

1. Problem Statement

Our implementation project will use a data set and Natural Language Processing model to classify the type of political bias present in a news article.

2. Possible Approaches

In this section, we will outline the possible approaches used for creating or retrieving a data set and choosing a Machine Learning Model, capable of Natural Language Processing. Before we begin, we must define the criteria used to identify possible approaches in retrieving data sets or models.
The ideal data set will contain either the URL or body of a news article, combined with a label for political bias. The political bias we are concerned with is left (Democrat) or right (Conservative). The labels used for these articles may be a multi-class label or some other quantifier suggesting bias.

Due to the nature of this implementation project, the NLP models discussed in Section 2.2 are preexisting, meaning that they have been developed by a third party organization and can be extended for a variety of NLP tasks, including political bias classification. The selection of this model will depend on how quickly the model can be setup, the complexity of its architecture, its hardware requirements, and the model's projected accuracy for political bias classification.

2.1. Data Collection

While there are many data sets for news articles, there are few that are labeled with a political bias. One data set [1] used crowd-sourcing to label news articles with a bias. The data set contains over 20,000 articles. Each article has a Democrat and Republican column, where the value in the column can be Positive, Somewhat Positive, Neutral, Somewhat Negative or Negative. This value represents the result of averaging the responses from a survey about the contents of the article, where they asked participants if the article was generally negative or positive toward their political party, either Republican or Democrat. The problem with this data set is that the bias is not represented in the format we expect (left, left-leaning, center, right-leaning, right), and instead is represented by two columns. It follows that we would have to create our own labels for each article based on the values from the Democrat and Republican columns. Another issue with the data set is that each article has not been pre-scraped, i.e only a link to the article is provided. To use this data set, we will have to scrape each article ourselves. Using the python library news-please [2], we can accomplish the task of scraping news articles, reducing the difficulty of the manual work.

The second data set we found has has a small and large form and was created for a research task at SemEval 2019 that pit teams against each other to build machine learning models that can detect hyperpartisan news (articles containing strong bias) [3]. The small data set contains 1273 articles manually labeled with a feature "hyperpartisan" that can be true or false. The larger data set contains 754,000 articles that are automatically labeled with a bias based on the publisher it came from. The bias can take the following values: 0 (right), 1 (right-center), 2 (least), 3 (left-center), 4 (left). The benefit of using these data sets is that all of the articles have been pre-scraped, meaning we would not have to manually scrape the body of the article ourselves. However, they do come with a few challenges. While the smaller data set has a great accuracy when it comes to labeling, the label itself is not ideally what we are looking for as it is a boolean label only specifying if the article is biased, with no indication of left or right. The large data set has labels that match perfectly with what we want, however the amount of noise in the data set is unknown. It is likely that many articles were mislabeled due to the assumption that every article from a publisher can be given the same label, which is surely false for many articles.

Another possible approach is to develop a custom data

set. This could be done by collecting articles posted to different communities within the popular social media platform, Reddit. Communities on Reddit are called subreddits and they are composed of content corresponding to the subreddit's topic. Combining the previously mentioned python library, news-please [2], and the Reddit API, we consider the following methods to scrape news articles from these subreddits:

1) Use the Reddit API to fetch URLs from political subreddits that we determine to be left or right leaning. This method works under the assumption that any news article posted in a specific subreddit will contain a political bias associated with the political stance of the subreddit. Therefore, we would label each article according to the subreddit it is posted in. This method may also limit the amount of noise in the data set as we can choose subreddits that primarily post political articles.
2) Use the Reddit API to fetch URLs from any news subreddit, and we assign a label based on the political rating of the publisher from Media Bias Rating [4]. This method works under the assumption that any one article shares the bias that is given to the news publisher by Media Bias Rating.

2.2. Model Selection

Our initial idea for what kind of model to use, was to select a pre-trained (i.e. prepackaged) model. This allows us to focus more on the implementation instead of spending time designing an architecture that works for our problem. Our expectations for a pre-trained model is that it is not computationally intensive or especially difficult to fine tune.

An analysis of pre-trained transformer models on detecting emotions in text was done on BERT-based models using the ISEAR data set [5]. The models analyzed were BERT, RoBERTa, DistilBERT, and XLNet. The results of the experiment concluded that RoBERTa performed the best and DistilBERT performed the worst. DistilBERT was computationally most efficient, followed immediately by RoBERTa. The researchers also concluded that XLNet and BERT are computationally demanding. Based on these results, the model's that interest us the most are RoBERTa and DistilBERT, due to the relative ease of fine tuning and comparatively low computational requirements. Access to these pre-trained models is simplified through the python library Hugging Face [6], which provides APIs for hundreds of pre-trained models, including DistilBERT and RoBERTa.

However, according to results obtained from the formerly mentioned research task at SemEval 2019 [3], using a pre-trained model may not be the best approach. Teams that used the pre-trained model BERT (with fine tuning) achieved low accuracy when compared to teams that used linear models or Convolutional Neural Networks (CNN). A different approach that some of the top performing teams used, was to use pre-trained models (like BERT) to compute word-embeddings, which would be used as features for a linear model or CNN. Teams that used this method received higher accuracy, therefore making the method a great candidate for our project. Although, this approach may be more difficult and time-intensive to implement.

3. Project Plan

Objective	Deadline
Create or retrieve data set.	Mar. 7
Create Flask App. Create UI elements (form to send URL to model)	Mar. 9
In Flask App, scrape data from the article given by URL. Have this data be ready to pass model.	Mar. 12
Implement model (Google Colab).	Mar. 12
Fine-tune model (Google Colab).	Mar. 19
Discovery: how to deploy model to Flask.	Mar. 19
Implement model in Flask App.	Mar. 26
Query the model from Flask. Show model's predicted label to user after they query the model with a URL.	Apr. 1

References

[1] C. Budak, S. Goel, and J. M. Rao, "Fair and Balanced? Quantifying Media Bias through Crowdsourced Content Analysis," *Public Opinion Quarterly*, vol. 80, no. S1, pp. 250–271, 04 2016. [Online]. Available: https://doi.org/10.1093/poq/nfw007

[2] F. Hamborg, N. Meuschke, C. Breitinger, and B. Gipp, "news-please: A generic news crawler and extractor," in *Proceedings of the 15th International Symposium of Information Science*, 03 2017, pp. 218–223.

[3] J. Kiesel, M. Mestre, R. Shukla, E. Vincent, P. Adineh, D. Corney, B. Stein, and M. Potthast, "SemEval-2019 task 4: Hyperpartisan news detection," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, 06 2019, pp. 829–839. [Online]. Available: https://www.aclweb.org/anthology/S19-2145

[4] AllSides, "Media bias ratings." [Online]. Available: https://www.allsides.com/media-bias/media-bias-ratings

[5] A. F. Adoma, N. M. Henry, and W. Chen, "Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2020, pp. 117–121.

[6] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6