Comparing Bag-of-Words, SBERT, and GPT-3 for Bias Detection

Max Luo[1] and Clayton Greenberg[#]

[1]Lynbrook High School, USA
[#]Advisor

ABSTRACT

This project aims to detect bias in media by training a machine learning model to recognize biased sentences. We did this by using a dataset containing 3700 sentences each annotated by multiple experts. The approaches we used were bag-of-words, SBERT, and GPT-3. For the bag-of-words and SBERT models, we generated prototype vectors for each class and used cosine similarity to classify sentences. For GPT-3, we used the OpenAI API's fine-tune function to train a model on the dataset, with the prompt being a sentence and the completion representing a class. The bag-of-words, SBERT, and GPT models achieved F-scores of 0.614, 0.819, and 0.838 respectively. We concluded that GPT-3 is the most accurate model while SBERT is the best model for a real-world application.

Introduction

Bias in media is inevitable, hard to discern, and interferes with readers' ability to formulate their ideas. Bias comes in many forms: omitting facts, emphasizing facts, wording, repeating certain facts, and more. This project focuses on detecting bias by word choice. Using a dataset annotated by multiple experts, a machine-learning model can be trained to recognize biased language. The models we tried either used sentence embeddings or generated the class the sentence falls into. The bag-of-words model generates sentence embeddings based on the number of word appearances, and the SBERT model uses a modified version of BERT to generate sentence embeddings. Part of our training data is allocated to generating prototype sentence vectors for each class, and we classify test sentences based on their cosine similarity to the prototype vectors. The generative model we used was GPT-3.

Background

There have been several studies on bias detection in the past. Spinde et al. (2021) is the closest to this study; we used the dataset they created. They trained BERT and BERT-based models to classify sentences as either biased or neutral, achieving an F-score of 0.804. Van den Berg and Markert (2020) tried to identify informational bias. They did this by including context from the entire article and articles on the same topic. Kameswari et al. (2020) created a classifier that detects political bias, focusing on the connection between presuppositions and bias.

Dataset

The dataset used for this project is SG2 from *BABE - bias annotations by experts* (Spinde et al., 2021)[1]. The dataset contains 3700 sentences from various news outlets, each of which is annotated by 5 experts. Each sentence is labeled

[1] Data can be downloaded here.

for a topic, political lean, biased words, bias rating (biased or unbiased), and expression of opinion (1-5). The final labels only contain the 3700 sentences with bias rating based on the majority vote of the 5 experts.

Metrics

The dataset is balanced in both bias rating and political lean. 49.3% of sentences are biased and the rest are non-biased. 26.9% of sentences lean left, 27.0% of sentences lean right, and the rest are either center or non-political. The dataset is also diverse: sentences fall into 23 different topics and are from 11 news outlets.

According to Spinde et al. (2021), SG2 has a Krippendorff's alpha (inter-annotator agreement) of 0.40, which is better than similar datasets.

Methods

We tried 3 different approaches: a bag-of-words model, SBERT, and a fine-tuned version of GPT-3.

Bag-Of-Words Model

The bag-of-words model is a simple model that counts the number of times each word appears in a sentence. This gives a vector with a length equal to the length of the vocabulary. We generated prototype vectors from the training subset for biased and non-biased sentences using their mean. Each sentence from the test subset is then classified based on its cosine similarity to each prototype vector. Words that do not appear in the training set are completely ignored during classification.

We chose to use CountVectorizer from sklearn (Pedregosa et al., 2011)[2] to generate the vectors. CountVectorizer has a few parameters that can be tuned. ngram_range changes the range of n-grams to be considered; for example, ngram_range=(1,2) will consider both unigrams and bigrams. Increasing this value will increase the accuracy of the model for large datasets, but higher values will result in extremely rare phrases. Our dataset is small, so n-grams beyond bigrams were not useful. Another parameter is vocabulary. By default, CountVectorizer will use the entire vocabulary of the training set. However, we can also specify a vocabulary to use: in our case, it could be the set of all biased_words. We decided to try both using all words available in the training set and using just the biased words.

SBERT

SBERT (Reimers and Gurevych, 2019) is a modification of the original BERT (Devlin et al., 2019) model. It uses a triplet loss function to train its Siamese architecture. This differs from BERT's cross-encoder architecture.

In practice, SBERT generates similar quality sentence-level embeddings compared to BERT, while being faster to train (Reimers and Gurevych, 2019). The SBERT website[3] includes a list of pre-trained models that are trained on large corpora of text. These models require no preprocessing; they take raw text as input and output a vector. In addition, they can be used directly or fine-tuned for our use case.

To fine-tune, we used the fit function included in SentenceTransformer. We entered the sentences and their label into InputExample's, which were loaded into a Dataloader and fed into the fit function. The fit function contains many parameters that can be tuned: we changed the number of epochs and loss function. To classify, we used the same approach as the bag-of-words model.

[2] CountVectorizer documentation

[3] Documentation here

To further explore the effect of fine-tuning, we tested an off-the-shelf model supplied by Google, the Universal Sentence Encoder (Cer et al., 2018). This model outputs 512-dimensional sentence embeddings that are intended to work well across many domains. As such, we did not fine-tune them.

OpenAI GPT-3

Generative Pre-trained Transformer (GPT) is a large language model that is most commonly used to complete text prompts (text generation). It has been adapted to perform chatting, inserting, and editing tasks. At the time of writing, GPT-4 is only accessible through a waitlist and does not support fine-tuning, making it unsuitable for this project.
We adapted GPT to perform text classification. To do this, we had to convert the data into a format that GPT can optimally process: a .jsonl file with columns prompt and completion containing sentences and their labels respectively. Since classes are recommended to be one token long, we converted the completion from "biased" and "non-biased" to "0" and "1" respectively. The data preparation utility from OpenAI also recommended that we add a space character before each completion, which we did.

We then followed this guide[4] to fine-tune GPT-3. Since all OpenAI models are closed and processing takes place on their servers, we did not have as much freedom to change GPT. However, two parameters that we did change were the base model (ada, babbage, curie, davinci) and the number of epochs. We used the ada (smaller) and curie (larger) bases and varied the number of epochs from 4 to 8.

To classify, we used the Completion.create function from the OpenAI Python API. The completions mimicked the completions we supplied in the training stage: a space character followed by a 0 or 1, corresponding to either non-biased or biased. In rare cases, the model output was not a 0 or 1. In this case, we could choose to assume a class, use another method of classifying the sentence, or consider the model output wrong for that sentence. We decided to assume the majority class (biased) for these sentences.

Table 1. Results

Model	Accuracy	Precision	Recall	F-score
CountVectorizer; ngram_range = [2,2])	0.642	0.647	0.593	0.614
CountVectorizer w/ biased_list vocab	0.587	0.566	0.600	0.582
Google Universal Sentence Encoder	0.693	0.717	0.688	0.702
SBERT: all-mpnet-base-v2	0.735	0.750	0.744	0.747
SBERT: all-roberta-large-v1	0.724	0.740	0.732	0.736
SBERT: all-mpnet-base-v2 fine-tuned, best parameters	0.823	0.828	0.811	0.819
GPT-3: curie-ft	0.833	0.841	0.835	0.838

[4] this guide

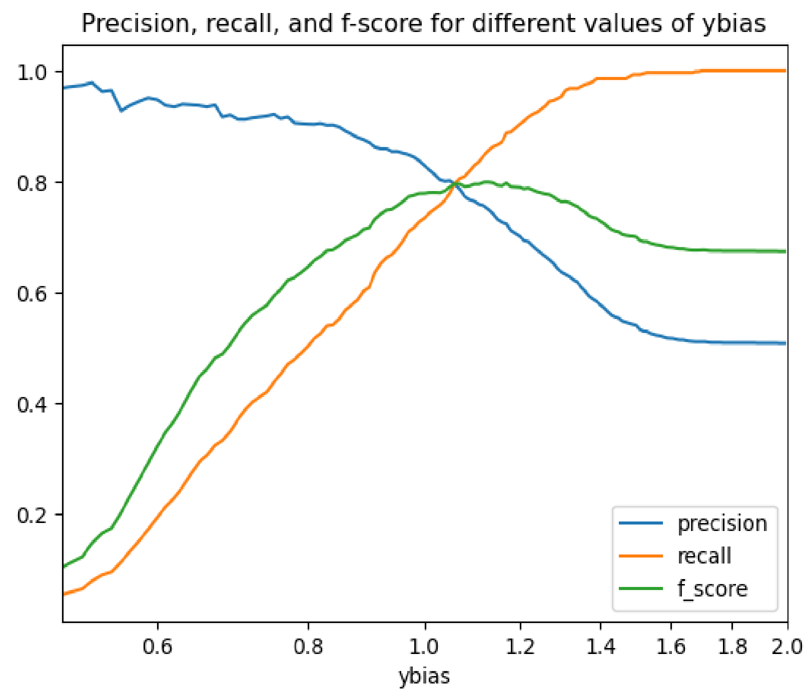

Figure 1. Metrics for different values of ybias in one of the fine-tuned SBERT models

Results and Discussion

As seen in Table 1, SBERT and GPT-3 far outperformed the bag-of-words model.

The bag-of-words model's F-score of 0.614 shows that there is some association between word appearances and bias, but it is not far off from the expected value for random guessing, 0.5. When we shifted from using all available words to just the biased words, the F-score decreased. We believe that this happened because the dataset is small and the list of biased words that happened to occur in this dataset is too small to generalize well.

Without fine-tuning (only trained on generic[5] data), SBERT outperformed the bag-of-words model by over 10% and the Universal Sentence Encoder by 4%. With fine-tuning, the model further improved by 7%. This shows that the similarity between biased sentences is related to traditional similarity, but differs in some ways. We also tried adding a multiplier, ybias, that increased the weight of biased sentences. The results can be seen in Figure 1.

GPT-3 performs slightly better than SBERT, with a 1-2% improvement in metrics. However, it is much more expensive to train and run. It is not open source and only runs on OpenAI's servers. Furthermore, we changed the formal task to accommodate a pipeline that comes with GPT-3. As such, the fine-tuning training data is highly unnatural and we need a lot of it for this pipeline to work. Therefore, we suggest that especially in a low-resource setting, the costs of using GPT-3 this way may outweigh the benefits and SBERT is the most practical model for a real-world application.

Conclusion and Limitations

We have shown three ways to classify text as biased or non-biased. The first is a bag-of-words model, which is the most primitive and least accurate. The second is a sentence embedding model, which performs very well and is the most

[5] pre-trained model information

practical. The third is GPT-3, which is the most accurate and demonstrates that generative models can be used to classify text.

All of these models have one major limitation: they only identify bias based on word choice. There are many other ways in which text can be biased, such as the omission or exaggeration of true information. The models are also incapable of identifying misinformation. While these articles' biases tend to show in their word choice, writers can write in a way that manipulates the reader without biased word choice.

A few simple ways to improve our results would be to use a larger dataset and to use an updated sentence embedding or GPT model. In addition, we are thinking about a program that rewrites articles to retain information but remove biases. Such a program could be made using GPT or another generative model. The model could be fine-tuned on a dataset of biased and corresponding non-biased text; while we are not sure such a dataset exists, we could create one by using the model itself.

A practical application of this project would be to build a news aggregator that assesses and informs the user of the bias of the articles they read. With a program that rewrites articles, the user could also read a single unbiased article that is written by AI, drawing information from many articles on the same topic. This would allow the reader to quickly get an uninfluenced overview of a topic.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess,
Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Lalitha Kameswari, Dama Sravani, and Radhika Mamidi. 2020. Enhancing bias detection in political news using pragmatic presupposition. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 1–6, Online. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR 2013*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Nils Reimers and Iryna Gurevych. 2019. SentenceBERT: Sentence embeddings using Siamese BERTnetworks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Timo Spinde, Manuel Plank, Jan-David Krieger, Terry Ruas, Bela Gipp, and Akiko Aizawa. 2021. Neural media bias detection using distant supervision with BABE - bias annotations by experts. In *Findings of the Association for*

Computational Linguistics: EMNLP 2021, pages 1166–1177, Punta Cana, Dominican Republic. Association for Computational Linguistics.
Esther van den Berg and Katja Markert. 2020. Context in informational bias detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6315–6326, Barcelona, Spain (Online). International Committee on Computational Linguistics.