Project Type: Implementation Project
Name: Sophia Lamphier
University of California, Berkeley (MIMS)
Course: Info_202

Training a Machine Learning Model for Bias Detection in Tweets

Introduction & Objective

The plan for this project was to create a new task in machine learning by defining a category of interest, annotating data, and training a machine learning model on it. For this implementation project, I collected and annotated all my own data. This project represents my first foray into machine learning, specifically focused on building a model to detect political bias in tweets on the social media platform, X. The goal was to classify tweets into two categories: "biased" or "unbiased" based on sentiment analysis. While accuracy was reduced due to my small dataset of 198 annotated tweets, the primary focus of this project was to learn and experiment with the ML workflow and tools used to execute. My broader aim was to empower users to evaluate the political bias of their trusted news sources, encouraging them to seek less biased information to form more well-rounded opinions and beliefs. I utilized TensorFlow's public tutorial videos and notebooks for assistance throughout the project.

Process & Methodology

1. Defining the Category of Interest

In light of recent political chaos, I thought it was only right to choose a topic that I am passionate about and that has a significant impact on our society; the spread of misinformation or bias. I was inspired by a research paper published by the Pew Research Center that looked into the news sources most utilized by particular political parties and how they have confirmed bias or misleading information (Mitchell, 2014). The task was to classify political tweets based on their bias orientation. Political bias in this context refers to content that intentionally or unintentionally supports a particular party (liberal, conservative) or takes a neutral stance. This categorization draws from our course discussions on grounded coding and the critical role of defining clear data categories in supervised machine learning tasks.

2. Data Collection and Annotation

I collected 198 tweets from the X platform (formerly Twitter) across a spectrum of political opinions. The data collection followed these principles:

- Diversity of Sources: Tweets were sampled from accounts known for left-leaning, right-leaning, and neutral political content based on the weight given by the Pew Research Center (Mitchell, 2014).
- Manual Annotation: Each tweet was labeled manually based on its tone, language, and content. Annotation followed a clear tagging guideline that I created inspired by class readings on transparency and accountability in dataset preparation.

The annotation process posed a few challenges including distinguishing between subtle biases and overt political messaging, which required iterative refinements to my labeling process. I was the only person to label my data which could potentially lead to un-inherent bias. In the future, I will think about crowdsourcing/getting annotation helpers which could help with the unintentional bias in labeling while also speeding up the process and gathering more data to help my model perform.

3. Data Preparation and Preprocessing

Preparation for training involved several preprocessing steps:

- Lowercase the text to standardize it.
- Remove unnecessary characters (e.g., punctuation, links, etc.).
- Convert it into a TensorFlow dataset for training.
- Shuffle, batch, and prefetch the dataset.
- Check a sample batch.
- Double-check the label distribution to ensure balance.
    - My dataset is pretty imbalanced which definitely caused my model to struggle. If I had more time, my next steps would be to oversample the minority class or use techniques like SMOTE.
    - 0 (biased): 113
    - 1 (unbiased): 85
- Tokenization: I did use a tokenizer during training, but instead of a traditional Tokenizer (e.g., from keras.preprocessing.text), I used a TextVectorization layer as part of my preprocessing pipeline. This layer acts as the tokenizer in my workflow.
    - Text Vectorization: Since TensorFlow operates on numeric data, I needed to convert the text into a suitable numerical format (e.g., tokenization).
- I did not account for overfitting.

4. Training the Model

The model development phase involved experimenting with different machine learning frameworks, including PyTorch and TensorFlow. I ultimately chose TensorFlow's framework because it was most intuitive and the help videos were abundant. I trained my model using the tf_dataset.

5. Evaluation and Fine-Tuning

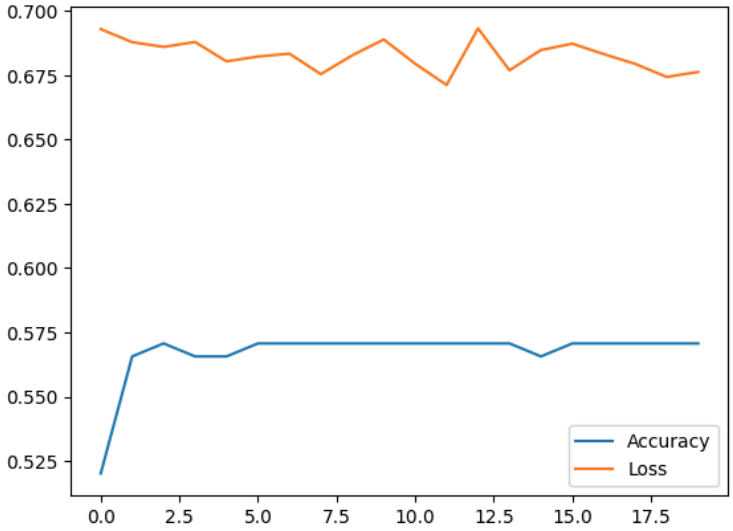
Evaluation metrics that were discussed in our course were used to understand my model's performance:

- Accuracy: I assessed the accuracy of the trained model by comparing the predictions it made to some notion of "truth" for those same data points.
    - My model's accuracy starts around 59% and fluctuates slightly during training, ending at ~58.8%.
    - Accuracy values around 50-60% suggest that the model is not learning effectively or is struggling with the data (probably because of how small the dataset is.

- Precision: The quality/ability of being correct.
- F1-Score: Balanced precision and recall to address the class imbalance inherent in the data

	precision	recall	f1-score	support
0	0.57	1.00	0.73	113
1	0.00	0.00	0.00	85
accuracy			0.57	198
macro avg	0.29	0.50	0.36	198
weighted avg	0.33	0.57	0.41	198

  - .
- Loss: A value that measures the difference between the predictions made by the model and the actual target values.



  - The loss metric starts at 0.6915 and reduces slightly to 0.6740 by the end of 10 epochs. This high and stagnant loss value probably indicates underfitting (the model is not learning enough to distinguish between the classes) which I mentioned I did not accompany for in training.
  - The accuracy and loss fluctuate significantly between epochs (e.g., dropping in some epochs and improving in others). This could be because of:
    - 1) Data Issues: Noise or imbalance in my dataset.
    - 2) Model Simplicity: The model architecture might be too simple for the task.
    - 3) Batch Size: (Most likely) A small dataset with batch processing could introduce variability.

6. Challenges in My Model:
- My dataset size was very small, which led to overfitting.
- I had limited differentiation between “neutral” and biased content, requiring further annotation refinements most likely due to the imbalance between the number of “0” and “1” datapoints in my dataset.

Class Concepts

The largest implementation of class concepts was through grounded coding when I was collecting and annotating my data. Inspired by class exercises, I developed a tagging guideline to ensure consistency, transparency, and accountability in my annotations. While I did not create a datasheet for my dataset as we did in a previous assignment, I now understand the value that they hold for understanding and utilizing the data contained.

In class 24, we discussed automatic classification and specifically, text categorization. This project was an attempt at a "supervised" machine learning algorithm that tried to find patterns that characterize human-defined categories. I gave the model training data in the form of <x,y> pairs (a binary text dataset) to then go learn h(x) with the space of categories being very small (sentiment analysis: positive/negative, 0/1). This manually labeled data guided the training process. My procedure echoed the flow discussed in this class: define categories, represent data, collect training data, train model, predict, evaluate, interpret.

Then when preparing my data for training, I used regular expressions to clean the data and then used a TextVectorization layer as part of my tokenization algorithm. My model utilized vectorization by converting the text into numerical format which was done by using the TF-IDF technique that we learned in class. I then tested my model with concept learned in class such as the F1-Score and accuracy. When my model was finished, I utliized my knowledge of Github from a previous assignment to host my code online for public access.

Reflection

The annotation process was incredibly time consuming and prone to subjective bias, so I will definitely look to get annotation help in the future. However, I found that transparent annotation guidelines helped me collect data more efficiently and ensured reproducibility and accountability.

While I didn't iterate very much, it is clear that iterative development and feedback loops are essential in ML workflows. I also learned that pre-trained models (like AWS) significantly reduce the barrier to entry for beginners in machine learning. Instead of trying to build something from scratch, you can use a pre-trained model and re-train it with the data you collected to produce a model.

Eventhough the accuracy of my model wasn't great, I still really enjoyed learning the process of creating a dataset, cleaning it, training a ML model on it, and testing. This project successfully taught me about the end-to-end machine learning pipeline and, with more time, I hope to produce a state of the art model one day that will really help people.

Bibliography

Mitchell, A. (2014, October 21). *Section 1: Media sources: Distinct favorites emerge on the left and right*. Pew Research Center. https://www.pewresearch.org/journalism/2014/10/21/section-1-media-sources-distinct-favorites-emerge-on-the-left-and-right/

https://www.tensorflow.org/tutorials