Data Science Techniques for Automatic Antisemitism Detection Nicolas Cloutier
Last edited February 13, 2023.

Difficulty of Rules-Based Classification

The initial takeaway I get from looking at the dataset is that is seems specifically tailored to be difficult for a rules-based system to classify. Many of the non-antisemitic tweets have words typically used by antisemites to express antisemitism and the vocabulary overlaps between the antisemitic tweets and the tweets that simply mention Judaism. There are a few words that appear to only come up in the antisemitic tweets, such as mentions of George Soros, but those words appear in few tweets and even in the case of filtering out mentions of Soros, that would also affect innocent mentions of him.

In the directory labeled "data", there is a subdirectory labeled "counters" that have files that coule be useful for comparing the frequency of different words in tweets that were identified as being different types of antisemitic. The file's title gives the type of antisemitism (a number 1 through 4, corresponding to political, economic, religious, or racial antisemitism), and in the file is a list of words next to a number. This number is the percent frequency of that word in the dataset of that type of antisemitism subtracted by the percent frequency of the same word in the non-antisemitic tweets, to give a view of which words were more or less common in these datasets. Looking at the words, it seems as if none of those that are significantly more or less common appear to be necessarily antisemitic words. As a result, only using the appearance of these words would likely not be very useful for classification. Instead, I decided to take a more holistic approach to the rules-based classifier.

Rules-Based Classifier Design

I begin by taking in an input text, putting it all in lowercase, removing stopwords, removing punctuation, and reducing it to a percentage frequency counter of each unique word in the text. For example, `"The big, big brown fox"` would be represented as {`"big": 50, "brown": 25, "fox": 25`}. This process is done for the combined text of all antisemitic tweets and non-antisemitic tweets, as well as individually for the tweets of each of the four types of antisemitism. I stored the resultant frequencies in a subdirectory of the "rulesbased-model" directory entitled "frequencies." In order to determine if a text is antisemitic, I do this same basic process to represent it as a series of frequencies. I then read the frequencies of the antisemitic and non-antisemitic datasets, removing any words from those datasets that do not appear in the input text and adding a new row with a frequency of 0 for words that appear in the input text but not the dataset. After that, I order the words alphabetically in all three `DataFrame` objects so they are all in the same order. Then, I store the second column of the `DataFrame` objects (the one containing the frequency percentages) in three vectors, one for the input, and the other two for the two datasets. I then find the distance between the input vector and each of the two dataset vectors, and choose whichever dataset has the smaller distance as the one that more closely represents the input data.

For example, for the counter {`"big": 50, "brown": 25, "fox": 25`}, the antisemitic dataset would have a counter {`"big": 0.056159, "brown": 0, "fox": 0`}, and the baseline would have {`"big": 0.125219, "brown": 0, "fox": 0.008348`}. The next step would be to sort these words alphabetically, but luckily the order they come in in the text is already alphabetical. Next, they would be represented as vectors and the distance would be found between them and the input text, as shown:

$$\begin{bmatrix} 50 \\ 25 \\ 25 \end{bmatrix} - \begin{bmatrix} 0.056159 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 49.943841 \\ 25 \\ 25 \end{bmatrix} = \vec{a}, ||\vec{a}|| \approx 61.19$$

$$\begin{bmatrix} 50 \\ 25 \\ 25 \end{bmatrix} - \begin{bmatrix} 0.125219 \\ 0 \\ 0.008348 \end{bmatrix} = \begin{bmatrix} 49.874781 \\ 25 \\ 24.991652 \end{bmatrix} = \vec{b}, \left|\left|\vec{b}\right|\right| \approx 61.13$$

Since the vector representing the non-antisemitic dataset is slightly closer to this input text, it would be classified as not antisemitic. The distances between these vectors becomes smaller as larger texts are inputted with frequency distributions that more closely match those of the datasets. An alternative method would be finding the cosine similarity between the vectors and choosing whichever has the highest as the closest distribution.

Model Evaluation

Since the model used distributions from the antisemitic tweet dataset, using these tweets for evaluation is not very useful. That being said, we can use it as a preliminary tool to make sure the model is identifying differences in word distributions. Testing the model on the all of the dataset tweets for the Boolean antisemitic classifications and the type classifications evaluated on only the antisemitic tweets for which classifications were provided, the following accuracy ratings (out of 1) were achieved.

Table 1: Evaluation on training data

Method	Task	Accuracy	Chance Accuracy
Vector distance	Boolean classification	0.76	0.5
Vector distance	Type classification	0.57	0.25
Cosine similarity	Boolean classification	0.76	0.5
Cosine similarity	Type classification	0.57	0.25

It would appear as if the method used has little effect on the accuracy, and that the models, scoring significantly above chance, have at the very least identified distribution differences in the tweets. The model can be used for binary classification on the 4Chan posts and normal tweets (a series of several thousand tweets randomly gathered from Twitter tweeted over the course of 2021) to check that they correctly identify that the 4Chan posts are mostly antisemitic and the normal tweets are mostly not antisemitic. Results of the model on those datasets are shown below. In Table 2, the "Bool. avg." column is the arithmetic mean of all of the Boolean classifications of each dataset, the "Cos. Bool. avg." column is the same but using cosine similarity, and the columns ending in "#" give the raw number of occurences of each type classification for each post in the dataset (using vector distance).

Table 2: Data summaries

Dataset	Bool. avg.	Cos. Bool. avg.	1 #	2 #	3 #	4 #
4Chan data	0.85	0.85	504	575	1474	793
Normal tweets	0.60	0.60	3675	1357	2419	3549
Labeled data	0.44	0.44	515	133	277	296

Again, the different methods for calculating similarity appear to not affect outcomes at all. The occurence columns are given with vector distance, but these data are exactly the same as those calculated using cosine similarity. Looking at the Boolean averages for the normal data, it appears to be higher than may have been anticipated or hoped for, given that the tweets were collected from a random sample of tweets across 2021. Interestingly, the Boolean average of the normal tweets is higher than the Boolean average of the labeled data, which contained antisemitic tweets.

Machine Learning Models

Three machine learning models were used: SVMs, Decision Trees, and Naïve Bayes classifiers. Each of those models was used for Boolean and type classification and with three seperate text representations (Bag-of-Words, raw frequency, and TF-IDF), making a total of 18 models. The evaluation metric used was the arithmetic mean of the recall for each individual classification in order to account for large class imbalances in the initial dataset. The models were then grouped by algorithm type and text representation and the most common answer was taken to find the differences between the classes. After this, a Cochran Q test was performed on the results, and, if they were significant, a post-hoc Dunn test was used to find the precise differences between the model types. This process was done for each task. The result of these tests can be found in the `cochran.ipynb` file in the `ml-models` directory.