Joey Ling
Professor Zili Liu
Psych 186B

Detecting Antisemitic Posts in Social Media

Introduction

Social media is a hugely beneficial tool and facilitates the circulation of different ideas, entertainment, and news. However, a recent surge of extremist hate groups in past years has made hate speech, and more specifically antisemitic speech, more and more common on all platforms. These seemingly harmless posts have had social implications and turned into action in the real world, as demonstrated through hate crimes and events like the January 6th insurrection on Capitol Hill. Online platforms are responsible for censoring hateful or inappropriate content, but many antisemitic posts cannot always be detected very easily. The growing number of cases of online antisemitism underscores an urgency to find better solutions for detecting hateful posts. Antisemitism is defined by the International Holocaust Remembrance Alliance (2024) as follows:

> Antisemitism is a certain perception of Jews, which may be expressed as hatred toward Jews. Rhetorical and physical manifestations of antisemitism are directed toward Jewish or non-Jewish individuals and/or their property, toward Jewish community institutions and religious facilities.

For our project, we fine tuned a large language model aimed at detecting and combating antisemitic content online. This has major implications in online censorship.

We started our project by reviewing existing research on the subject. Our initial plan was to replicate the model created by Chandra et al. (2021), who created and trained models using BERT and RoBERTa to detect antisemitic tweets on Twitter. However, due to issues with the Twitter API and tweet-scraping, we had to change our approach. We decided instead to adopt an already established AutoNLP model for antisemitism detection from HuggingFace, fine tune it, and train it (training one from scratch is infeasible as it would require too much time and money). Our goal for this project, which we achieved, was to be able to fine tune and retrain a model such that it could accurately classify social media posts that are antisemitic.

Datasets

For our project, we used two separate datasets: the ISCA dataset and SBIC dataset. The former comes from the Institute for the Study of Contemporary Antisemitism, and it contains around 7,000 tweets with around 18% of them being antisemitic. The latter comes from the Social Bias Inference Corpus, and it contains over 100,000 annotated offensive posts from Reddit and Twitter. Both datasets have a column containing the post text and a column containing a value of 1 for an antisemitic post or 0 for a non antisemitic post. We mainly used the ISCA dataset for all three of our models, and we used the SBIC dataset to help us further explore each model.

Understanding Transformers

Our project used pre-trained, transformer-based large language models, which are deep learning model architectures used mainly for natural language processing (Hugging Face). These models have been trained on lots of raw text and weigh the importances of different words in a string (Hugging Face). The architecture of a transformer consists of an encoder, which receives an input and builds representations of the features, and a decoder, which helps generate the outputs from the representations

(Hugging Face). A transformer works by first tokenizing a string (breaking it down into words and assigning it an integer value), creating embedded vector representations (as mentioned in the encoder definition), and allowing us to fine tune the model with our own parameters and datasets to get the predictions (Hugging Face). In our case, we used a transformer and fine tuned our model using Pytorch and the Trainer API from Hugging Face.

Methodology

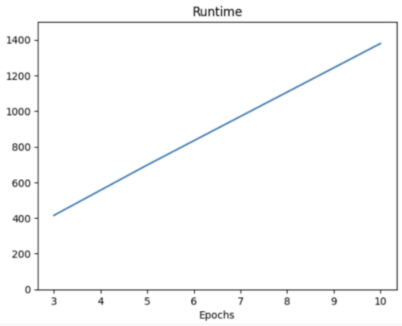
We first imported important libraries, such as transformers, sklearn, and torch. Then, we loaded in the datasets and preprocessed our data. For our models, we only needed two columns: Text and Label, so we removed any unnecessary features from each dataset, such as a post ID or date of the post. Afterwards, we tokenized our text, which consisted of separating strings into individual words and assigning them an integer value based on a pre-set vocabulary attached to our model. Finally, we loaded the model, which was an antisemitism model adopted from Hugging Face that had previously already been trained on a different dataset (Starosta). Next, we tried different methods and fed in different data sets to fine tune our model and improve the predictive power of our model. After fine tuning and training, we tested our model on various datasets and made modifications in antisemitic posts to measure how well our model could classify antisemitic texts.

Model 1: Using the Trainer API

The Trainer class from Hugging Face is helpful as it can automatically train, evaluate, and fine-tune transformer based models for users . It creates an automatic training and evaluation loop in PyTorch, so users don't have to implement as much, but it doesn't provide as much control or transparency over the training and evaluation process. For this model, my partner and I used the default arguments, which included a dynamic learning rate and 3 epochs.

Figure 1

Runtime over Epochs



Model 2: Manual PyTorch

Although the first model provided easy implementation and fine tuning, we felt that it was too much of a "black box," such that we didn't truly see code for implementation and fine tuning. We were consequently motivated to manually conduct a full training of our model using pytorch. In this model, we used 5 epochs to test for overfit or underfit. We originally used 10 epochs, but due to a lack of time and processing units, we had to reduce the number of epochs to 5. Figure 1 shows how much runtime increased with the number of epochs.

Once we set the number of epochs, we also set the batch size to 8 and the optimizer to AdamW. AdamW is a method that comes from stochastic gradient descent, but it also includes weight decays (Keras). Next, we set the learning rate to dynamic, which was initialized at 1.0×10^{-7}. For each epoch and each batch within that, we then computed the output and loss. The output was measured by logits, which are raw unnormalized predictions that come from the model (Hugging Face). The logits that came from our predictions were 2 dimensions, as our classification task is a binary task (Hugging Face). We then predicted whether a post was antisemitic or not based on the max value in those dimensions.

Model 3:

We were curious about whether our model was beneficial, since it targets antisemitism specifically versus any general hate speech. To examine this, we also trained and fine tuned a third model from Hugging Face, which was trained to detect any hate speech. We used all the same parameters as Model 2: 5 epochs, dynamic learning rate, batch size of 8, and AdamW optimizer.

Results:

We will discuss the results of Models 1, 2, and 3. We measured the performance of each model on the testing set after 5 epochs of training using three metrics: accuracy, sensitivity, and loss. Accuracy is the percentage of overall correct predictions, meaning both true positives (ground truth is 1 and prediction is 1) and true negatives (ground truth is 0 and prediction is 0). Sensitivity, also known as recall, is the number of true positives out of ground truth positive instances. We measured loss using mean square error. Finally, we also compared runtime to see how long it takes for the models to converge.

The results for each model are shared below. Compared to the original model that we adopted from Hugging Face and did not fine tune, Models 1, 2, and 3 all had a higher accuracy and a lower loss. Thus, by fine tuning and retraining our models, we were able to improve the performance of our models. However, there was also a lower sensitivity. As a result, we wanted to explore different ways to improve or worsen the models. We conducted all of our testing using Model 2.

Figure 2

Comparison of All Models

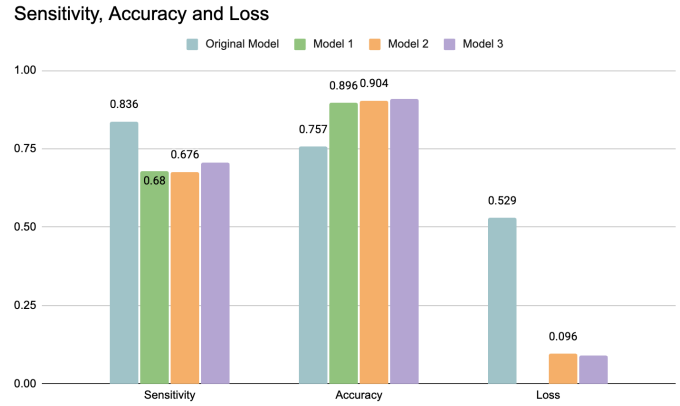


Table 1

Comparison of All Models

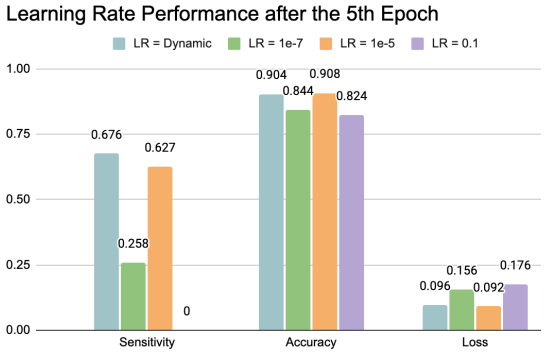
	Original Model	Model 1	Model 2	Model 3
Sensitivity	0.836	0.68	0.676	0.705
Accuracy	0.757	0.896	0.904	0.909
Loss	0.529		0.096	0.091

Note. Original Model is the model adopted from Hugging Face which we later used for fine tuning

Exploration/Breaking the Model: Tuning Hyperparameters

We first tuned hyperparameters to analyze how performance varies. The first hyperparameter was learning rate. With a dynamic learning rate, a scheduler initializes the learning rate to 1.0×10^{-7}, and after each optimization step, it adjusts to a new learning rate between the initialized value and 0. The performance here was the same as our original Model 2. We compared its performance to different static learning rates: 1.0×10^{-7}, 1.0×10^{-5}, and 0.1. The performance of static rates differed greatly. For the first rate, the accuracy and

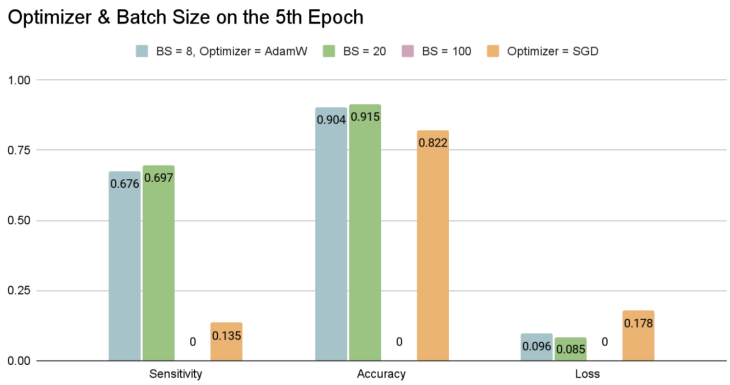
Figure 3



sensitivity both increased very steadily, though it ended after 5 epochs with metrics still lower than with the dynamic learning rate, thus demonstrating how it takes too long to converge. The second rate was slightly better, but still varied a bit. Finally, the last learning rate performed significantly worse, with its accuracy fluctuating between 0.2 and 0.8. In Figure 3, we show the performance on the test set after 5 epochs, using different learning rates during training.

Next, we tried different batch sizes: 8, 20, and 100. The first batch size was tested on our original Model 2. With a batch size of 20, the performance was similar, but runtime significantly increases to 808 seconds. To save time, we stuck with the batch size of 8. When the batch size was 100, our program failed to run due to a lack of RAM.

Figure 4



Note. "BS = 8, Optimizer = AdamW" represents the baseline Model 2, which we used to compare the batch size and optimizer. Values for BS = 100 are initialized to 0 because the program crashed.

Finally, we tested on a different optimizer. The AdamW optimizer is often used to train deep neural networks with large datasets and tends to converge faster. The stochastic gradient descent optimizer requires more manual tuning. Using an SGD optimizer made our model perform significantly worse, decreasing the accuracy by nearly 10 percent. Its metrics slowly improved over time, but not fast enough to perform better than when we used the AdamW optimizer.

Table 2

Comparison of Tuning Hyperparameters

	LR = Dynamic	LR = 1e-7	LR = 1e-5	LR = 0.1	BS = 8	BS = 20	BS = 100	Optimizer = AdamW	Optimizer = SGD
Sensitivity	0.676	0.258	0.627	0	0.676	0.697	0	0.676	0.135
Accuracy	0.904	0.844	0.908	0.824	0.904	0.915	0	0.904	0.822
Loss	0.096	0.156	0.092	0.176	0.096	0.085	0	0.096	0.178
Runtime	762	739	739	691	762	808	0	762	678

Through tuning various hyperparameters, we found that the best performance was with a batch size of 8, a dynamic learning rate, and the AdamW optimizer. Any other changes made to the model tended to worsen the accuracy, sensitivity, and loss, and therefore could be said to break the model.

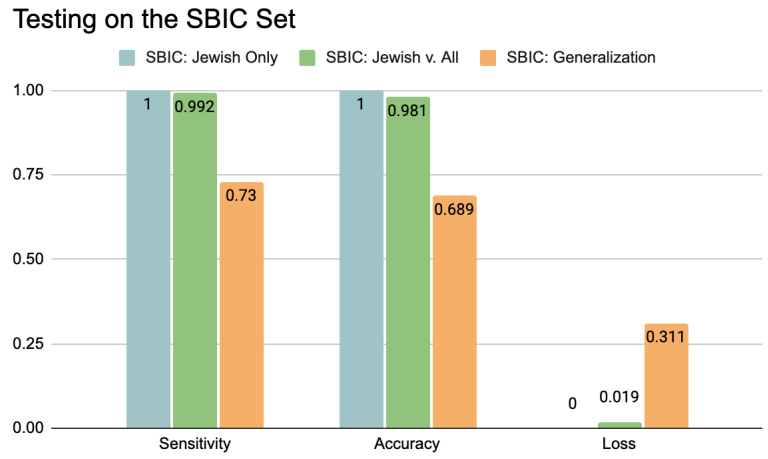
Exploration/Breaking the Model: SBIC Dataset

Next, we wanted to test the model's fine tuning and parameters on a different dataset. We retrained and tested the model on 3 different datasets that consisted of nearly all hate speech. We wanted to test whether the model could distinguish between any hate speech and specifically antisemitic speech.

The first test ("SBIC: Jewish Only") consisted of feeding in a dataset that only contained antisemitic texts. Every entry was labeled as 1 (antisemitic), and the performance was perfect because we had essentially trained our model to always predict 1. The second test ("SBIC: Jewish v. All") consisted of feeding in a dataset that contained all hateful speech, but only antisemitic texts were labeled as 1 and the rest were labeled as 0. We had to manually preprocess this part and modify the entire dataset in order to do this. We expected the performance for this test to be slightly worse than the first test, since we had a more balanced dataset. However, it only decreased very slightly, which suggests that our model is well

trained in distinguishing hate speech from antisemitic speech. Because the SBIC dataset comes from much more extreme and explicit language, we hypothesized that it was this type of language that made it easy to distinguish. Finally, we tested our SBIC model's ability to generalize to any hate speech at all ("SBIC: Generalization"). We preprocessed and filtered through the entire dataset once again to change any hate speech to have a label of 1 and non hate speech to have a label of 0, regardless of whether or not it was antisemitic. The texts labeled as 0, although not technically hate speech, still contained foul language and inappropriate content. We expected our model to perform worse since it is generalized to all hate speech now, and we were correct. The accuracy went down to around 0.69.

Figure 5



Exploration/Breaking the Model: Modifications to our Test Sets

After receiving feedback for our presentation, we conducted extra exploration to try to understand our model further. For the final exploration, we modified the texts to see if our fine-tuned model would be able to perform well on test sets that contained drastically altered text. We used the baseline Model 2 for training, and replaced the test set with texts that were modified. We conducted three major manipulations on the test set: scrambling, removing words like "Jew," and removing antisemitic slurs.

First, we scrambled the text. This was done by shuffling the words in each post, then preprocessing and tokenizing. An example of a shuffled text would be: "I love Psych" → "Psych love I." The accuracy decreased, but only slightly to 0.857. On the other hand, the sensitivity increased to 0.79, which suggests that scrambling the words actually improved the model's ability to classify a ground truth antisemitic post as antisemitic.

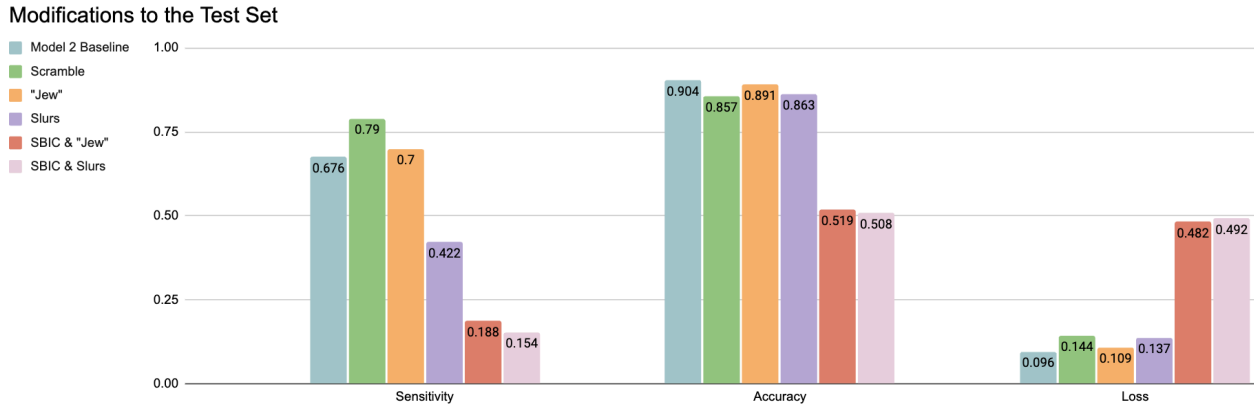
We tried to explore this further by unscrambling the text and removing the word "Jew" from all posts in the test set. We also made sure to remove other related words, and the complete lexicon of removed words is as follows: ["Jew," "Jews," "Jewish," "Judaism"]. An example of this modification would be: "I am **Jewish**" → "I am." Surprisingly again, the sensitivity was still higher than our baseline Model 2. The accuracy was still lower (0.891) than the baseline Model 2, but by a very small margin, and it was also higher than the accuracy for scrambled words. This suggests that our dataset may contain antisemitic texts that are quite subtle and not very easy to detect, and that removing the word "Jew" may not impact performance very much because the remaining content of the post still indicates the presence of antisemitism. This is positive because many social media posts are not very

explicit and can contain very subtle hints of antisemitism. Because our sensitivity was still high (0.700), we were still able to flag antisemitic posts. This means that in a less explicit post, we can still detect antisemitism.

Additionally, in order to ensure that our model was not only detecting the presence of slurs, we censored posts by removing slurs from the text. The slurs were contained in a file called "lexicon.csv," which is attached in our project submission. An example of this modification would be: "I don't like **antisemitic slur**" → "I don't like." This significantly lowered the sensitivity of our text to 0.422, meaning it couldn't flag the antisemitic posts as well anymore, but still left the accuracy relatively high at 0.863. We inferred that our model either performs best when there are slurs in the text, or that posts with slurs simply contain less text in general and therefore less context pointing to an antisemitic post (which would leave a post with little to no context once a slur was removed).

In order to test our hypothesis, we duplicated these tests, except on the SBIC dataset. The SBIC dataset contains more dark and offensive jokes in general, and it contains much more explicit language, even on the posts that are not considered antisemitic. We hypothesized that the model would perform worse on this set because it would not be able to tell that a post was antisemitic without slurs. Our results below show that the model performed significantly worse, scoring an accuracy just above chance on both modifications (removing words from the Jewish lexicon and the slur lexicon). The sensitivity significantly decreased, suggesting that our model failed to flag many antisemitic posts. This is in line with removing slurs from the original dataset, and indicates that posts with explicit slurs tend to carry less extra content indicating antisemitism. Once the slurs are removed, there is less context that can indicate the presence of antisemitism. This can be harmful, as our goal is to flag as many antisemitic posts as possible. However, the fact that our model still performed well for the first dataset suggests that we can still flag antisemitism in less explicit posts.

Figure 6



Note. Model 2 Baseline represents our Model 2 which was manually fine-tuned and retrained using PyTorch. Scramble represents scrambling the test set text. "Jew" represents removing words from the list of neutral words representing Jews. SBIC & "Jew" represents the same manipulation, except done on the SBIC test set. Slurs represents removing slurs contained in lexicon.csv. SBIC & Slurs represents the same manipulation, except done on the SBIC test set.

Exploration/Breaking the Model: Testing Model 3 on SBIC

We wanted to test if there was any benefit to using an antisemitic-specific model over a general hate speech model, but based on our initial results, Model 3 (model for detecting general hate speech) had performance comparable to Model 2. As a result, we used the SBIC dataset, which contained a large

amount of hate speech, converted half of the posts to be labeled as 1 for specifically antisemitic posts, and then converted the other half to 0 for hateful or non hateful speech that was nevertheless non-antisemitic. This is the same procedure as the plot for “SBIC: Jewish v. All" in Figure 5. The performance was much lower in this instance and had an accuracy of only 0.513 and recall of 0.071, which suggests that Model 2 is great for detecting specifically antisemitic text. Model 3, while good at detecting any hate speech in general, cannot distinguish antisemitic text from hate speech. Thus, Model 2 is preferred in this context.

Conclusion

In summary, we trained and tested 3 different models, and we used different methods to tweak each model and understand what its strengths and weaknesses were. We first fine tuned a model adopted from Hugging Face and ran it with the Trainer API (Model 1), but we didn’t like that it didn’t show us much about the training loop. This led us to conduct a manual training using PyTorch (Model 2). We used Model 2 to test our hyperparameters and found that the best hyperparameters were the following: batch size = 8, learning rate = dynamic initialized at 0.0000001, optimizer = AdamW. We then tested on a different, more explicit and offensive dataset (SBIC), and found that performance was still very high. It also had fairly good results for general hate speech. Afterwards, we manipulated the texts in both our original dataset and the SBIC dataset, and we removed words like “Jew” in one test and antisemitic slurs in another test. We used the trained Model 2 and tested on these manipulated texts. We found that in general, the model can still perform relatively well on a dataset with moderate language, but that it performs worse on datasets that contain lots of explicit language. This suggests that the presence of slurs is a large indicator of inappropriate content, but that even on posts without highly explicit language, our model still works well. Finally, we compared Model 2 to Model 3, which was a general hate speech detection model adopted from Hugging Face. We found that Model 3 can perform well but falls short when trying to distinguish between hate speech and specifically antisemitic speech. Thus, Model 2, the one that we spent the most time modifying and exploring, works best.

Overall, we have fine tuned and retrained a model that can detect antisemitism better than general hate speech models and can still detect antisemitism in the absence of the word “Jew” or antisemitic slurs. Through our work, we have modeled a strong solution for social media platforms to censor inappropriate content and subsequently lower the spread of hate speech that can carry over into the real world.

References

Behind the pipeline. Hugging Face. (n.d.-b). https://huggingface.co/learn/nlp-course/chapter2/2?fw=pt

Chandra, M., Pailla, D., Bhatia, H., Sanchawala, A., Gupta, M., Shrivastava, M., & Kumaraguru, P. (2021, June 18). *"subverting the jewtocracy": Online antisemitism detection using multimodal deep learning*. arXiv.org. https://arxiv.org/abs/2104.05947

Facebook. (n.d.). *Facebook/Roberta-hate-speech-dynabench-R4-target*. Hugging Face. https://huggingface.co/facebook/roberta-hate-speech-dynabench-r4-target

Fine-tuning a model with the trainer api - hugging face NLP course. Hugging Face. (n.d.). https://huggingface.co/learn/nlp-course/chapter3/3?fw=pt

A full training - hugging face NLP course. Hugging Face. (n.d.). https://huggingface.co/learn/nlp-course/en/chapter3/4?fw=pt

How do transformers work? - hugging face NLP course. Hugging Face. (n.d.). https://huggingface.co/learn/nlp-course/chapter1/4?fw=pt

Institute for the Study of Contemporary Antisemitism @ Indiana Universtiy. (n.d.). *Isca-iub/antisemitismontwitter*. Hugging Face. https://huggingface.co/datasets/ISCA-IUB/AntisemitismOnTwitter

Jikeli, G., Karali, S., Miehling, D., & Soemer, K. (2023). *Antisemitic Messages? A Guide to High-Quality Annotation and a Labeled Dataset of Tweets*. ArXiv. https://arxiv.org/pdf/2304.14599.pdf

Sap, M. (n.d.-a). *Social Bias Frames*. Maarten Sap. https://maartensap.com/social-bias-frames/

Sap, M., Gabriel, S., Qin, L., Jurafsky, D., Smith, N. A., & Choi, Y. (2020, April 23). *Social bias frames: Reasoning about social and power implications of language*. arXiv.org. https://arxiv.org/abs/1911.03891

Starosta, A. (n.d.). *ASTAROSTAP/autonlp-antisemitism-2-21194454*. Hugging Face. https://huggingface.co/astarostap/autonlp-antisemitism-2-21194454

Keras Documentation: ADAMW. Keras. (n.d.). https://keras.io/api/optimizers/adamw/

Trainer. Hugging Face. (n.d.). https://huggingface.co/docs/transformers/main_classes/trainer

What is antisemitism?. IHRA. (2024, February 9). https://holocaustremembrance.com/resources/working-definition-antisemitism