Can Machines Detect Hate Speech in Memes?

WARNING! This post contains content that is highly offensive

The Hateful Memes Challenge

Artificial Intelligence plays a crucial role in mitigating problematic or abusive behavior, such as pornographic material, misinformation, and hate speech on social media platforms. While machines may be good at classifying content that is just images or just text, memes pose a unique challenge because it is often the combination of image and text which make a meme hate speech or not. For example, the words “polish remover” might not automatically be considered hateful, but they certainly are in the image displayed below.

Training a machine to recognize this as hate speech is not as easy as it sounds. While humans can easily integrate the text and image information holistically, machines struggle with this sort of task. Currently it is impossible for humans to inspect all the content posted on social media sites so being able to create a machine learning model which can accurately categorize memes as hate speech is becoming ever more important. With that in mind, the goal of our project was to develop a machine learning system that could determine whether a meme was hateful or not.

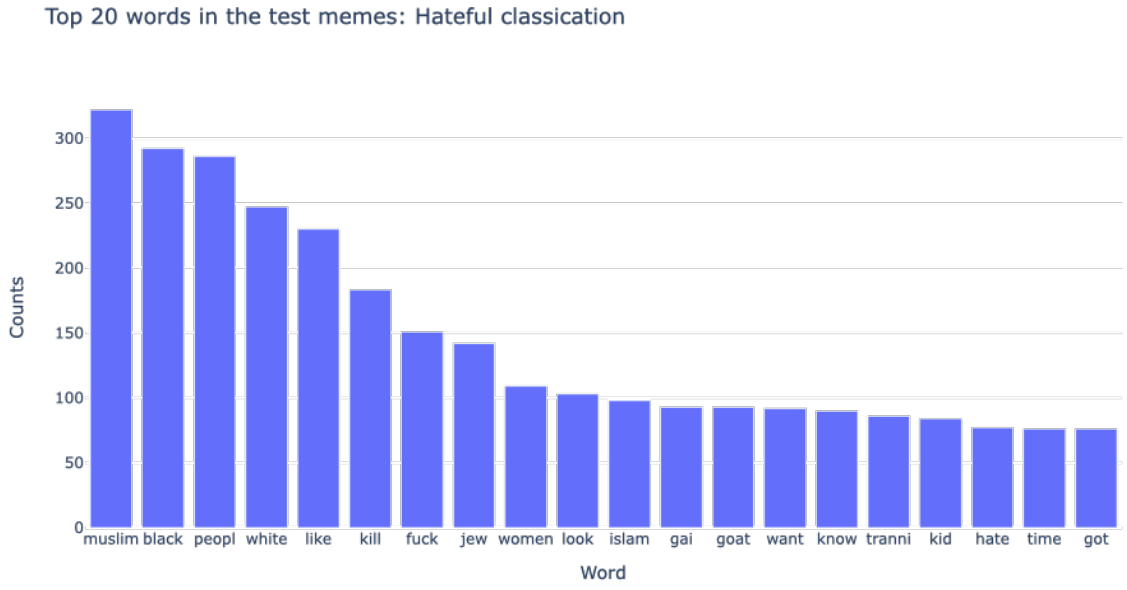
In order to achieve our goal we used the Hateful Memes Challenge dataset which was created by Facebook AI. This dataset includes a training set of 8,500 memes as well as a validation dataset and a hold out test dataset. All three datasets include the images as well as a JSON file with the meme text and label. According to the article *The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes* the memes were created by Facebook and classified by third-party annotators who used the below definition of hate speech to classify the memes:

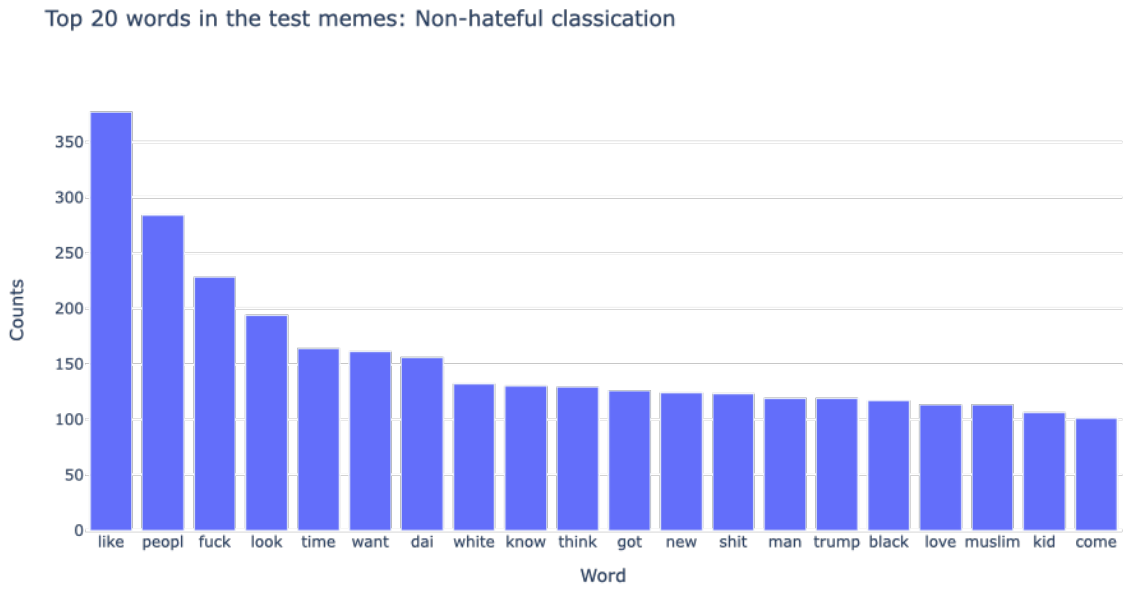
> *"A direct or indirect attack on people based on characteristics, including ethnicity, race, nationality, immigration status, religion, caste, sex, gender identity, sexual orientation, and disability or disease. We define attack as violent or dehumanizing (comparing people to non-human things, e.g. animals) speech, statements of inferiority, and calls for exclusion or segregation. Mocking hate crime is also considered hate speech."*

Each meme was classified by three different annotators who were trained on how to classify the memes. The annotators could give a score of 1-3 with 1 indicating "definitely hateful", 2 indicating "not sure" and 3 indicating "definitely non-hateful." Each meme was classfied five different times with the final classification being decided by majority rule. In order to ensure the memes could not be classified by simply using just text or just images, Facebook also included "benign confounders" which were created for the memes annotated as hateful. Benign confounders were defined as "a minimum replacement image or replacement text that flips the label for a given multimodal meme from hateful to non-hateful" (Kiela et al, 2021).

When looking at the training memes, approxmately 36% were classified as hate speech. When looking at the text of the hateful memes versus the non-hateful memes there did seem to be some differences in the top 20 words with words relating to race, gender, and religion being more prevalent in the hateful memes. This gives some indication of how the meme text will influence the classifier.

Meme Examples





What has already been done to solve the problem?

While a lot of research has been done on unimodal models less has been done on multimodal models, especially when it comes to classifying memes. According to the article *The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes* "There has been surprisingly little work related to multimodal hate speech, with only a few papers including both images and text." One of the main reasons Facebook set up this challenge is to expand our understanding and research into this area. According to documentation from the Hateful Memes Challenge there are a couple different ways this has been approached in the past. With regards to data representation, there are really two ways to fuse the data; early fusion and late fusion. In early fusion systems the text and image embeddings are combined prior to classification while in late fusion systems a classifier is built just based on the text and just based on the image information and then the results from those two systems are combined to determine the final classification. Prior research has shown that early fusion systems tend to perform better than late fusion systems (Kiela et al., 2020). Besides early or late fusion systems multimodal models are often based on pretrained models (Kiela et al., 2021). Transfer learning from pretrained models has been shown to work well with multimodal models. These models can be unimodally pretrained or multimodally pretrained. In unimodally pretrained models two unimodally trained models are combined in some way, for example, combining a BERT model and a ResNet. In multimodally pretrained models the model was pretrained using a multimodal objective. Examples of pretrained multimodal models include VilBERT and VisualBERT.

Since this challenge was completed earlier in 2020, we had the advantage of being able to see how the winners completed the task and what their results were. The winners of the challenge were able to achieve an AUROC of 0.85 on the unseen test data. According to the paper, *Enhance Multimodal Transformer With External Label And In-Domain Pretrain: Hateful Meme Challenge Winning Solution* the winners used an ensemble method which incorporated four of the most state of the art multimodal models: VL-BERT, UNITER, VILLA, and ERNIE-Vil. They also extracted other features from the images including the context of the image as well as race and gender if there were people in the memes. Clearly this solution is effective but very complex. We wanted to see if we could take some of the strategies learned from previous research into this area, such as early fusion and pretrained models, as well as some of the techniques used by the winners, like including race and gender, to create a simpler but effective multimodal model. Prior to creating our multimodal model we wanted to get a performance baseline so we decided to train two other unimodal classifiers: one that just uses the image and one that just uses the text. Similar to the challenge, we used AUROC as our test metric since this gauges how well a classifier separates the two classes.

Ethical Implications

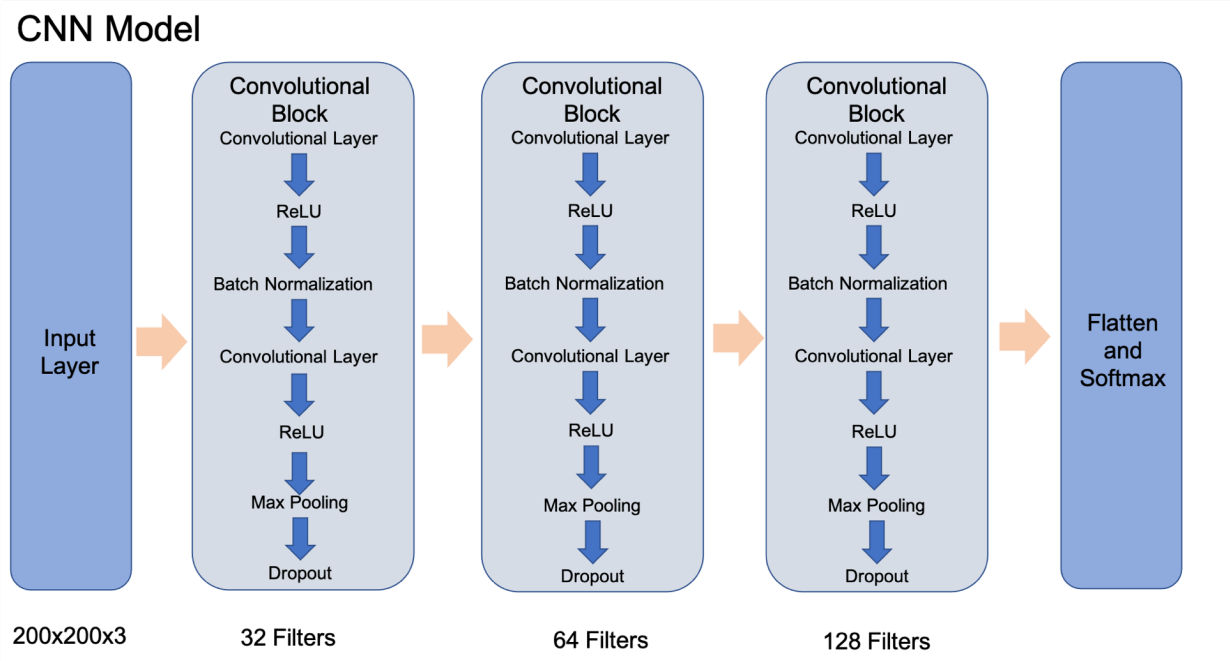
Clearly any type of model which could be used to censor speech has serious ethical implications. If we create a model that is too sensitive it may censor memes which are not actually hateful. This could lead to other consequences for users such as temporary or permanent bans from the platform. On the other hand if a model is not sensitive enough it might prove to be useless and therefore subject people to highly offensive content. While these are definitely important ethical considerations something else to keep in mind is what are the biases of the annotators who initally classified the memes? Obviously Facebook tried to make the process more objective by providing a defintion of hate speech as well as providing training for how to classify the images but in many ways deciding if something is hate speech is inherently subjective and colored by each persons experiences, beliefs and culture. For example, the use of derogatory terminology towards certain ethnic groups is not always considered hate speech when used by a person of that ethnic group. Depending on the background of the human annotators they may or may not be able to pick up on these nuances. Obviously there are still biases if you use human content inspectors, but formally building these types of biases into an AI system can be very dangerous. You would need thousands of content inspectors, each with their own biases, to go through the same content that one machine learning model, with one set of biases, could go through. The fact that Facebook only used three annotators for each meme and provides no information about them makes us weary of using any model in a production setting that is soley trained on this data. This is especially true given the fact that no analysis was done by Facebook or the challenge winners to look at bias. In our opinion they should have used a larger number of annotators, provided some general information about them, and conducted a bias analysis. One of our hopes when taking on this challenge was to be able to provide an analysis of bias in the training data but unfortunately time got the best of us and we were not able to

complete this task. With this in mind we created our multimodal model with the sole purpose of furthering research in this area and not to be used in production on any social media platforms.

Image Classification Using a Convolutional Neural Network

What did we do?

The first thing we tried was to create a classifier simply based on the image data. In order to do this we trained a Convolutional Neural Network since this form of model can capture the spatial information in images. CNNs are usually comprised of several convolutional layers, nonlinear activation functions, and pooling layers. A diagram of our model can be seen below.



As you can see the input layer consists of images which are 200x200x3. CNNs require that all the input data is the same size but unfortunately since we are dealing with real world data the images were all different sizes. In order to deal with this we first used cv2 to resize the images.

When it came to the actual convolutional layers one of the parameters we had to decide on was the kernel size. The kernel is what slides over the image to create a hidden layer, or feature maps, and includes the learned weights. We decided on a 3x3 kernel since this tends to be the go to size for most CNNs. As for the nonlinear activation function, you can see we decided to use ReLU. Nonlinear activation functions help the model learn complex

relationships within the data. ReLU does this by either returning a value in a feature map directly or zero if the value is less than zero. One of the main advantages of using this activation function over another is that it helps to simplify the model and speed up the learning process since some values are returned as zero. We also included batch normalization which normalizes the output of the first convolutional layer. This pushes the values in the hidden layer to have a mean of 0 and a variance of 1. Similar to normalization in a regular model, this helps to speed up learning and can also help to regularize the model. Next we added another convolutional layer before then adding a max pooling layer with a size of 2x2. The max pooling layer takes the feature maps from the previous convolutional layer and finds the max value in each 2x2 area. This also helps to regularize the model and reduce overfitting. Lastly we use a dropout layer with a probability of 0.1 for dropping any of the features in the previous layer. We set the dropout to 0.1 because we found the model tended to underfit. We still wanted to keep a dropout layer to prevent overfitting but set it lower to help with the underfitting problem. The model repeats this convolutional block three times with a different number of filters. We used a different number of filters in each convolutional block since the number of filters corresponds to the number of feature maps and what parts of the images the model learns. After applying three convolutional blocks we then flattened the output and applied a softmax function in order to get the classification predictions. For optimization of the model we used cross-entropy loss since this is popular to use with binary classification deep learning models as well as the "Adam" optimizer.

When it came to training the model we first made sure to normalize the data between 0 and 1 in order to speed up the training and hopefully lead to a more accurate model. We also implemented early stopping to again hopefully avoid overfitting. We focused our early stopping on the validation AUROC since this is the metric we used to evaluate our models and set the patience to five which means after five successive training epochs where the validation AUROC is less than our best AUROC the model will stop training. We decided on a patience of five since three seemed to stop the training too early and anything more than that seemed to take a very long time to train without providing any benefit. We also implemented a checkpoint so that we could save the model with the best validation AUROC. Lastly, we decided on the number of training epochs and batch size. We chose 30 as the number of epochs because this is large enough where the model should be able to learn a lot from the training data but not so many epochs where the model takes forever to train. For the training batch size we decided on 32 since smaller batch sizes often lead to better generalization on the hold out test data (Chang, 2020).

How did it work (or not work?)

Unfortunately this method did not seem to work. Usually with deep learning models you worry about overfitting but we actually had the opposite problem; the model never seemed to learn enough to even make good predictions on the training set. Even with the best model we only achieved an AUROC of 0.5 on the training data which means the model was not able to differentiate the positive and negative classes on the training data. The AUROC on

the hold out test set was slightly worse at 0.48. We also looked at the accuracy and found the model accurately classified 51% of the memes. When we looked at how the classifier was making predictions we found that it was simply predicting all of the memes as non-hateful speech. In the examples below, the label indicates what class the model predicted. If the label is in green that means the prediction was corret, if it is in red the prediction was incorrect.

Not hate speech

Not hate speech

Not hate speech

Not hate speech

Not hate speech


Not hate speech

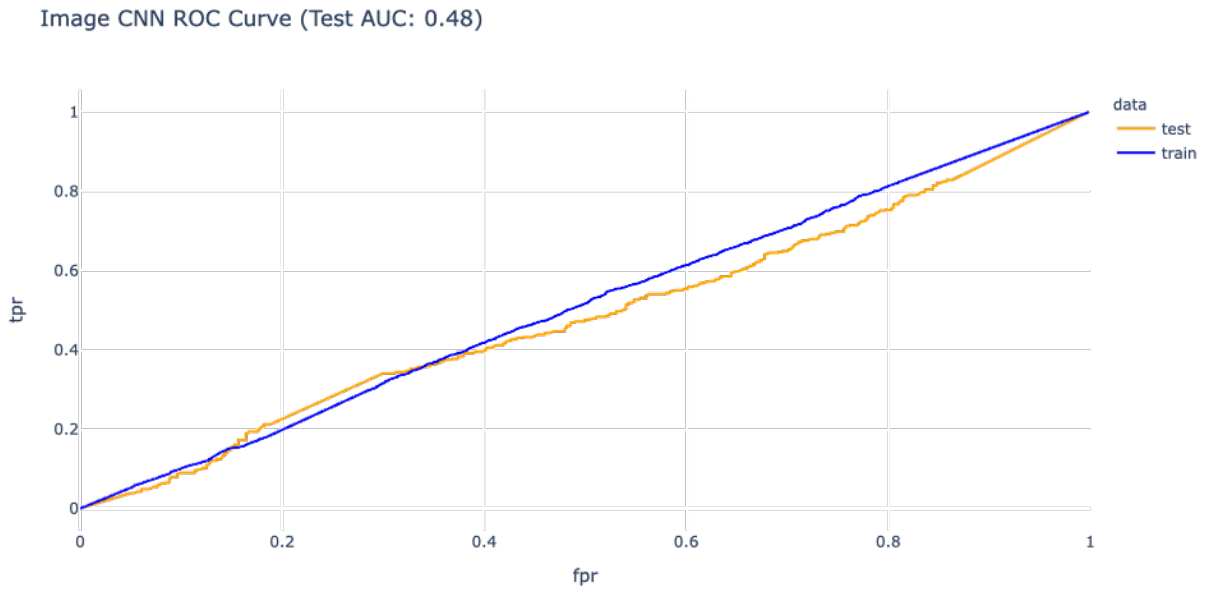
Not hate speech


Not hate speech

Not hate speech

Not hate speech

The reason for this probably has to do with the inclusion of those "benign confounders" since there were images which were represented in both the hateful and non-hateful classes in the training data. This makes it impossible for the model to make reliable classifications. The training data also included more non-hateful memes which probably contributed to the problem as well. Due to the "benign confounders" and unbalanced training data we hypothesized that these are the results we would achieve but still wanted to build an image only model as a baseline to compare against our multimodal model.



BERT Model

What did we do?

Another model we decided to try was a pretrained BERT model using only the text from the memes. BERT stands for Bidirectional Encoder Representations from Transformers. According to the *BERT - transformers 4.7.0* documentation "unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers." We decided on this method since it has been used in the unimodally trained multimodal models and tends to perform very well on a multitude of natural language problems. Specifically we used 'BertForSequenceClassification' and the 'bert-base-uncased' pretrained model since our goal was to use the sequence of text to create a classifier. Since this is a pretrained model there were only a few parameters we had to tune such as batch size, optimization function, and number of training epochs. This can be one of the benefits of using a pretrained model since it comes pretty much ready to run with very little work. Similar to the CNN model we chose 32 for our batch size. We also decided on AdamW for our optimizer since it tends to lead to better generalization (Graetz, 2020). Lastly, we decided to train for 30 epochs since we initially found that the model was underfitting so we wanted to see if training for longer would improve our resuts.

How did it work?

Using this method we were able to achieve an AUROC of 0.60 on the hold out test set and 0.75 on the training dataset. While this shows the model is able to differentiate the two classes better than the image only model it does not work nearly as well as the model from the challenge winners. When we looked at how accurate the model was, it was able to accurately classify 61% of the memes. Looking at a breakdown of how accurately it classified each category we found that it accurately classified only 35% of the hateful memes and 86% of the non-hateful memes. Obviously the text only classifier does a much better job at classifying the non-hateful memes.

Not hate speech

Not hate speech


Not hate speech

Not hate speech

Not hate speech

Hate speech

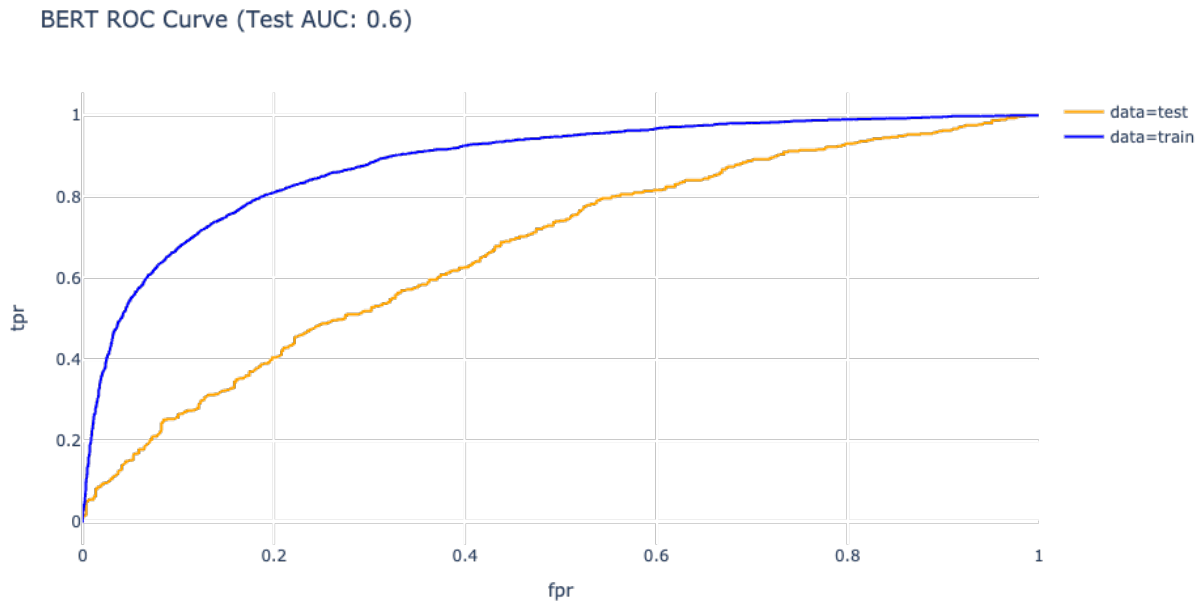
Not hate speech


Not hate speech


Hate speech

Hate speech

Similar to the image only model it is still underfitting since even the AUROC on the training set is not very high. Unfortunately training for more epochs only led to overfitting so this was the most balanced model we trained. We were pleasantly surprised though that this model worked better than the image only model. This is an indication that the meme text is probably more important than the meme images when it comes to classifying them as hate speech.



Multimodal Classification Model

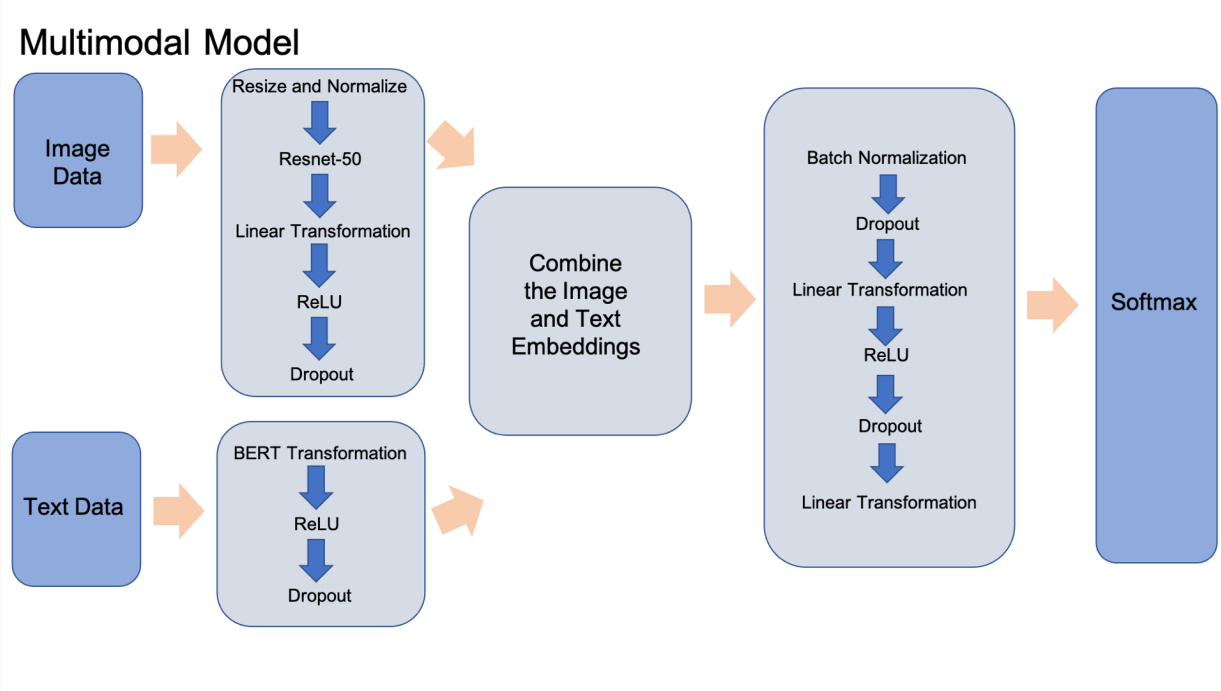
What did we do?

Lastly, we created our multimodal model using an early fusion feed forward neural network. Prior to using a feed forward neural network we tried to implement a more state-of-the-art solution using VisualBERT which is part of the HuggingFace library. According to the documentation, it is "a simple and flexible framework for modeling a broad range of vision-and-language tasks. VisualBERT consists of a stack of Transformer layers that implicitly align elements of an input text and regions in an associated input image with self-attention". While VisualBERT itself seemed simple to implement, part of the process required building a custom detector/feature extractor to get the visual embeddings. Rather than using the whole image, VisualBERT requires the embeddings to only represent objects in the image. We attempted to do this using a library called detectron2 but found the process confusing, time consuming and computationally costly so unfortunately we had to abandon this approach and instead focus on the feed forward neural network approach.

The first thing we had to decide on was how we were going to embed the text and image information. Since we got some positive results using BERT with our text only classifier we decided this was the best representation for the text data. We specifically used the Sentence Transformer library since it is "a modification of the pretrained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings" (Reimers & Gurevych, 2019) This fit our task since we needed embeddings to represent the full text of the memes and we wanted them to be semantically meaningful. We

specifically used the "paraphrase-mpnet-base-v2" pre-trained model because it was trained using a multitude of text data and had the highest average performance of any of the pre-trained SentenceTransformer models on several different natural language processing tasks. Next we decided to use a pretrained ResNet to embed the image data. We decided on this feature representation because, similar to BERT, this is a cutting edge way of representing image data. ResNets are able to accurately represent images without the drawbacks of using a deep convolutional neural network, like difficulty of training. Specifically we used the resnet50 which uses 50 layers. We also tried using resnet152 which uses 152 layers but this led to overfitting. Prior to obtaining the ResNet embeddings we also resized the images to 224x224, transformed them to a tensor and then normalized the tensor.

After determining our feature representation we then focused on the model architecture. Below we have provided a diagram of the model.



As you can see the model consists of some linear layers, ReLU activation functions, batch normalization, dropout, and a softmax function. Similar to when training our CNN image classifier we were concerned with overfitting which is why we included several dropout layers as well as batch normalization and ReLU. With the dropout we also used 0.5. Initially when we started training the model we tried 0.1 as well as only using dropout once but found that the model severely overfit (AUROC on training data was 0.99 but only around 0.52 on the hold out test set). We also added batch normalization in later training experiments to help with our overfitting problem as well as removed one of the linear layers but removing the linear layer actually led to the model severely underfitting.

Results

The best AUROC we could achieve on our hold out test set was only 0.61 with an accuracy of 60%. When looking at the model's performance on each class, it was able to accurately predict 52% of the hateful memes and 68% of the non-hateful memes.

Hate speech

Not hate speech


Hate speech

Hate speech

Not hate speech


Hate speech

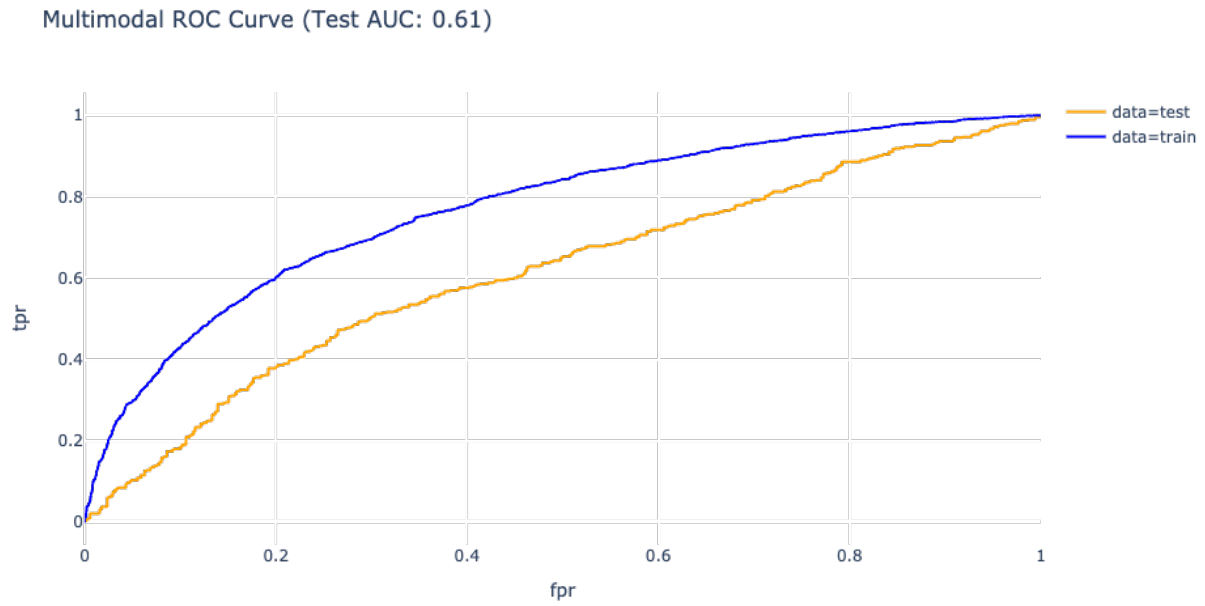
Hate speech

Hate speech

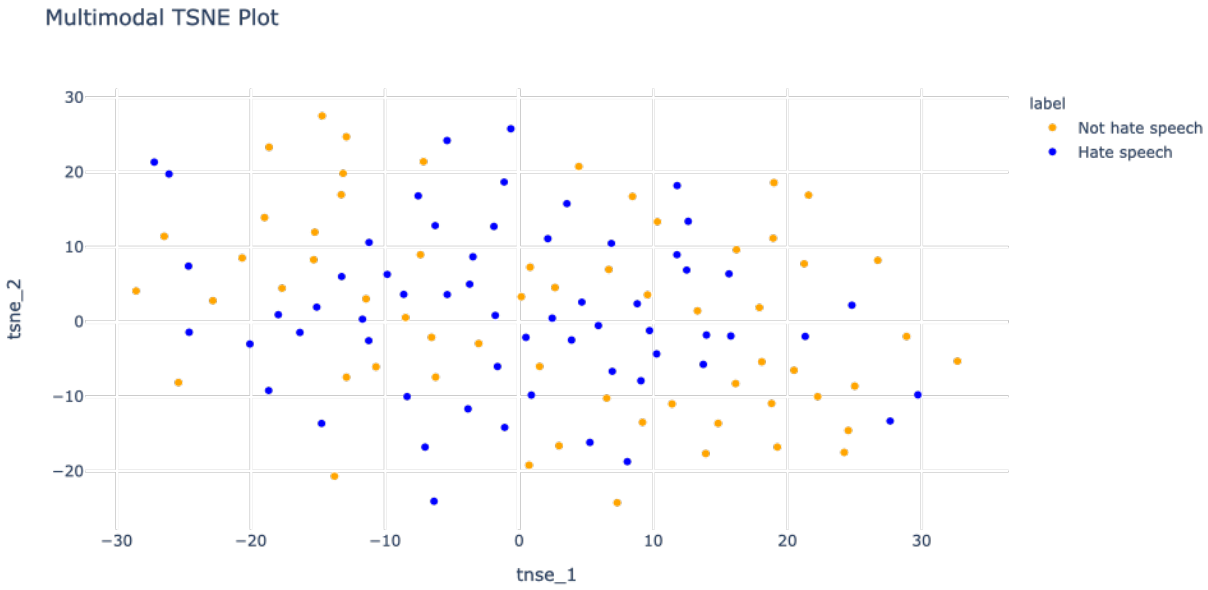
Hate speech

Not hate speech

While this is not great, it is still better than the results we achieved using only the image or text information. When comparing the multimodal model to the text only model, they both achieve about the same AUROC and accuracy score but the multimodal model is much better at classifying the hateful memes. This shows that it really is the combination of text and images that make a meme hateful. Unfortunately, this model is also underfitting a bit since we only get 0.77 AUROC on the training data. This indicates the model is not learning enough of the important information to perform well on either the test or training data. As noted before, we tried several different techniques to get the model to fit well but unfortunately they either led to very poor performance on the hold out test set while still overfitting on the training set or they led to underfitting on both datasets. In our opinion this means that our actual data representation or type of model may not be suitable to solving this problem.



Another indication that the data representation may not be suitable is demonstrated in the image below which shows a random sample of 60 hate speech memes and 60 non-hateful memes plotted using TSNE. As you can see, there is no separation between the classes when we plot the data in two dimensions. Although this is a rather dramatic reduction in dimensionality, it does give us an indication that using the ResNet and BERT embeddings may not work great.



It is very possible that tweaking either the data representation and/or type of model may have provided better results. Overall, there are several ways we could have improved upon our current model.

What would we do differently?

Given more time we would have liked to try some other techniques to optimize our current model as well as see if we could get the VisualBERT model to work. With the current model we balanced the classes in the training data by downsampling but we think a better technique may be to actually upsample the hateful memes in order to balance the classes since adding more training examples can improve model performance. One way we could have done this is by using Synthetic Minority Oversampling Technique, aka SMOTE, which synthesizes new training examples from the minority class. Along with upsampling, testing out different data representations may have been helpful. For example, with our text embeddings we used a specific pretrained BERT model but it is possible another BERT model trained on different data would have performed better. We also would have liked to spend more time implementing a grid search to optimize our training parameters in a more systematic way since this may have revealed parameters which led to a better fit model. Lastly, we would have liked to include information about the race and gender of human subjects in the memes. This is something the winners of the Hateful Memes challenge said really helped to improve the performance of their model since many of the hate speech memes deal with race or gender. While we attempted to programatically extract this information using the same method as the challenge winners, unfortunately, we were not able to get it to work reliably and add this information to our model. In the future this is something we would like to add in the hopes of improving the performance.

We also wish we had more time to implement the VisualBERT or VL-BERT methods since according to the Hateful Memes documentation, as well as the winners of the challenge, these methods would provide better results. These models are specifically designed to be used with visual and text data so it makes sense they would perform better on this task. We are hoping to implement a solution using this technique in the future and possibly try an ensemble method using our feed forward neural network and the VisualBERT model since several of the best performing models in the challenge used ensemble methods.

Conclusion

So can machines detect hate speech in memes? Yes...but it is hard! While our multimodal model did not perform as well as we had hoped we believe that our efforts demonstrate this is a complex problem without a simple solution. Even the challenge winning model could only accurately classify 73% of the memes while humans were able to accurately classify 85%. Obviously further research is needed in this area but we are proud to have made a contribution to a domain that all of us feel very passionate about!

Sources

BERT - transformers 4.7.0 documentation. (n.d.). https://huggingface.co/transformers/model_doc/bert.html (https://huggingface.co/transformers/model_doc/bert.html).

Chang, D. (2020, August 12). Effect of batch size on neural net training. Medium. https://medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57 (https://medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57).

Graetz, F. M. (2020, February 12). Why adamw matters. Medium. https://towardsdatascience.com/why-adamw-matters-736223f31b5d (https://towardsdatascience.com/why-adamw-matters-736223f31b5d).

Kiela, D., Firooz, H., & Mohan, A. (2020, May 12). Hateful Memes Challenge and dataset for research on harmful multimodal content. https://ai.facebook.com/blog/hateful-memes-challenge-and-data-set (https://ai.facebook.com/blog/hateful-memes-challenge-and-data-set).

Kiela, D., Firooz, H., Mohan, A., Goswami, V., Singh, A., Ringshia, P., & Testuggine, D. (2021, April 7). The hateful memes challenge: Detecting hate speech in multimodal memes. https://arxiv.org/abs/2005.04790 (https://arxiv.org/abs/2005.04790).

Reimers, N., & Gurevych, I. (2019, August 27). Sentence-BERT: Sentence EMBEDDINGS using siamese bert-networks. https://arxiv.org/abs/1908.10084 (https://arxiv.org/abs/1908.10084).

VisualBERT - transformers 4.7.0 documentation. (n.d.). https://huggingface.co/transformers/model_doc/visual_bert.html (https://huggingface.co/transformers/model_doc/visual_bert.html).

Zhu, R. (2020, December 15). Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution. https://arxiv.org/abs/2012.08290 (https://arxiv.org/abs/2012.08290).

GitHub

You can find all of the code we used for this project here: https://github.com/roseandgold/HatefulMemesProject (https://github.com/roseandgold/HatefulMemesProject)

Statement of Work

Laura Stagnaro: Preprocess Meme Image and Text Data, Preprocess the Images for CNN, CNN Image Model, attempted the VisualBERT model, Multimodal Model, Exploratory Data Analysis, BERT Model, blog post, GitHub repo

Jakob Cronberg: attempted the Race and Gender Extractor, blog post, video

Anel Nurkayeva: (special contributor): BERT Model, blog post