CS7643 Final Project: Hateful Memes (Memes2Vec)

Sandeep Arisetti
sarisetti3@gatech.edu

Marc Evans
mevans306@gatech.edu

Tyler Lang
tlang35@gatech.edu

Abstract

Memes have grown as a popular form of communication in digital social networks, and have been adopted as a way of popularizing shared opinions, with one use-case involving transmitting hateful messages, which social media companies would like to be able to classify and prohibit. However, memes provide modern challenges to traditional forms of deep learning and classification due to their bi-modality, subtle language cues, the contextualization that the image provide to the text, their hidden cultural references, and their high variability in content. As a solution, we experimented with various methods of solving the bimodal problem such as image captioning transformer models, and then using NLP models like BERT and roBERTa. We also experimented with creating a uni-stream and dual-stream model, in which the image and text are encoded together or apart. While we were unable to beat current state-of-the-art bimodal models such as Facebook's own MMF model, we were able to decipher the various challenges in meme classification, and propose further techniques for improving existing models via ensemble methods. We also explore image pre-processing techniques, such as object detection and text masking.

1. Introduction/Background/Motivation

To address the bi-modality of the memes, we attempted three distinct methodologies: (1) Image captioning and object detection models that uses image-to-text encoders to translate the image to text, which could then be fed alongside the meme's text in an NLP model such as roBERTa; (2) Using standalone encoder models for the image and text and run these results through a final transformer, and (3) use a single encoder that encodes the text and image together, such as VisualBERT, and pass the encodings through a transformer trained on bimodal data structures, such as CLIP. As this specific issue of detecting hateful memes does not yet have a widely agreed-upon solution, we felt it necessary to explore these different methodologies to understand the strengths and limitations of each, and inspire future research in the field.

Current methodology involves various techniques. Winning-code uses very specific pre-trained models on race, skin tone, and object detection to then pass on to models like visualBERT and ERNIE, and uses a voting method across multiple models' outputs to decide on final classification. Other techniques involve using convolutional neural networks to extract features from the images, and pass these features along with encoded text to a final dual-stream model. While the use of such specific feature extraction methods such as race classification and skin tone might perform very well on more direct and race-based hateful memes, we feel as though such models might suffer on more subtle representations of hate within a meme, especially when non-human entities are used in the image to transmit the hateful message. Thus, we aimed to score comparable accuracy using more generalized models, such as models pre-trained on text sentiment or image object detection.

If successful, the result of such efforts would directly aid in creating safer spaces in the internet, if adopted by social media platforms. While our techniques ultimately were not able to beat the current best-performing methodologies, we expect our efforts to be continued with further research in both hateful meme detection and bimodal classification tasks in general.

For this project, we used a dataset specifically curated for this task, involving over 10,000 images of both hateful and harmless memes, which was provided by DataDriven. This dataset includes the raw meme images, the text extracted from the meme, the classification label (1 for hateful, 0 for harmless), and the defined train,validation, and test set meme IDs.

2. Approach

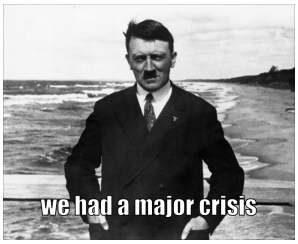
We started with removing captions from the images using an Optical Character Recognition (OCR) model in Keras, to identify and then remove the letters in each image, to then feed into our single and multi-stream processes. The model produces bounding boxes around the words and use a mask on each word occurrence to "fill in" the space using the Naiver Stokes method. Refer **Figure 1** that shows the original image and after removing text.

Then, we took an exploratory approach in which we used

(a) Original Image

(b) After running through OCR

Figure 1: Example of Text Removal



(a) important figures go unrecognized, which can make a difference in interpretation

(b) Objects and image don't portray the context of the text

Figure 2: Importance of detecting sentiment and important people

already-published code that takes a uni-stream approach via a CLIP model, in which the model is pre-trained to pair images and text together. With this code, as a way to learn more about the data itself and inspire our own analyses, we ran it with various alterations, such as changing the data source the model is pre-trained on, the learning rate and decay, the optimizer, and the loss function. The published code specifically uses a pre-trained model trained on hate speech, with the original author's intention being that regular hate speech would be the most representative to the text seen in the memes. We, however, felt that memes have a very unique lexicon and syntax that could make a highly specialized training dataset fail, and thus we experimented with both a model pre-trained on general bert-uncased data, a model pre-trained on meme sentiment data – which we felt might perform the best – and a roBERTa model trained on sentiment analysis from twitter data. The original code uses a binary cross entropy with logits loss function (BCEWithLogitsLoss), which makes sense given the binary label, but, for experimentation, we also attempted using a standard cross entropy loss function, which failed to optimize. Furthermore, we tried using an Adam optimizer, over the original code's use of Madgrad, which saw similar results as Madgrad.

After learning especially about how different pre-trained models versus architectures affect, or don't affect, validation accuracy and AUC, we decided to then create our own architectures to approach the problem, and learn further about how architecture itself affects model performance. To start, we designed a dual-stream model, which encodes image and text separately, using a pre-trained text encoder and an image feature extractor. Despite our feeling that such a model will ultimately perform poorly due to potentially losing the interaction effect between the text and image, in which the meaning of one is dependent on the other, we still felt there was a potential for such a model to independently learn image features that might otherwise be lost when the image encoder is too contextualized by the text, such as potentially capturing features of famous historic figures which the machine might learn to recognize, despite the text it is paired with.

Inspired by human cognition and light informal research we conducted on how memes are deciphered by humans – in which we asked friends and family to respond to certain memes in the dataset, explain their interpretation of the meme, and how they arrived at deciding if it is hateful or not – we aimed to develop an architecture in which images are translated to text, and this text, along with the text on the meme itself, are encoded together in a uni-stream BertForSequenceClassification model. One of the difficulties in interpreting memes is defining the relationship between the image and the text. In some cases, the image provides sentiment, in which the object or persons in the meme are not of importance, but their presence, or the sentiment they portray, either augments or adds irony to the text.

In other cases, the image serves as a contextual reference to the text, often containing subtle references that might be difficult to understand by the non-target audience.

Thus, our intention with this architecture is that, if we assume there is a model that can accurately describe an image via text to represent the features that a human interprets from an image, such as emotions, objects, and context, then such text can be most easily compared to the meme text with a single encoder. While we assume this type of model might work best, the biggest difficulty in this was thus finding an adequate textual representation of each image. We started by using an object detection model,in which we'd pass objects as words into the encoder. Due to time and memory constraints, however, we were forced to use only pre-trained models such as Hugging Face's DETR architecture, all of which detected only the most basic representations of objects, such as "person" or "sheep," without being able to focus on smaller details, such as clothing types, or

objects in the background or foreground. We tried a zero-shot object detector as well, but this still required being fed labels, which we did not have an exhaustive list of.

Abandoning object detectors, We first tried to train our own image captioning model based on a flickr dataset, which provided captions much too general for our dataset. We then used a pre-trained image captioning model from Hugging Face, which resulted in much better captions, in which captions were generated using "nlpconnect/vit-gpt2-image-captioning" pre-trained image captioner, which uses an instance of VIT for image encoding and GTP-2 for decoding via causal language modeling. It provides a faster, "greedy" method, and a slower and more descriptive "un-greedy" method. Still, we quickly realized that, for many memes, the cultural references themselves are often the most important content of the meme image, such as displaying a particular political or historical figure, which captioning cannot parse, such as in figure (2-a).

Finally, we tried another version of a uni-stream model, using a pre-trained VisualBERT model, using early fusion. Single-stream or early fusion is a classification technique that uses a multi-modal model with 2 or more modal inputs (such as images and text) to classify. After failures with dual-stream modeling, we deduced that single-stream would be more effective, because of the subtle context between text and visual data, especially in the case where both are neutral by themselves, but in context produce a hateful, or positive message. This is especially true when considering sarcastic, or otherwise oxymoronic content that requires an understanding of the context surrounding them. Our initial theory was – if you could look at the visual embeddings and text embeddings for an (image, text) pair, the distance between embeddings would signify how closely aligned the pairs were to each other. This could indicate sarcasm where the distance was large, or congruency if the distance was small, signifying sentiment could be a good indicator. As we explored this further, we experimented with the huggingface sentiment model, but this only produced a single probability score on positive vs. negative sentiment – and was unable to detect sarcasm.

Because of this, we looked at multiple multi-modal models including VisualBert. The approach with VisualBert was to first create the visual embeddings by using detectron2 (which implements object detection algorithms) and mask R-CNN. To create the embeddings, we had to first extract the features at multiple layers of complexity using a pre-trained checkpoint using mask R-CNN on COCO dataset. From these features we generate a "proposal" which are potentially important regions of the image features. We then use these proposals and features to draw bounding boxes, and filter down those bounding boxes to the most relevant ones, which we define the values of those boxes in the feature space as our visual embeddings.

3. Experiments and Results

During the course of our experimentation, we used two main metrics to measure the success of a particular model: the accuracy and the AUC.

3.1. Published CLIP Model (Basis for Evaluation of other Architectures)

As this published code served primarily as a baseline and stepping stone for our exploration of models, we ran the following experiments to test whether optimization choices affect model performance: using pre-trained models trained on general BERT uncased data, twitter sentiment data,and meme sentiment data; increasing and decreasing the learning rate and decay rate; changing to an ADAM optimizer; and using standard cross-entropy loss in exchange of Binary Cross Entropy with logits. The following accuracies and AUCs are recorded in Table 1. Note that an original model was ran with the original images, in which the meme text is part of the image. The same model was then run on our images with masked text, and from there on, all accuracies and AUCs refer to models run on the images with masked text.

As can be seen, the best model is ultimately the original model published by the author, in which an AUC of 0.73 is achieved. We were able to achieve just slightly better by using our pre-processed images in which the text is masked out, with an AUC of 0.74. We were surprised that these images didn't improve results further, but this also shows the limitations of the architecture of the model might ultimately be capping the success of the model. Our hypothesis by using images with the text is that any vision-based feature extractors would pick up false features in the white block text, affecting accuracy. Given that isn't the case, we assume that the vision encoder is not able to pick up on the finer details of the image in general. Our hypothesis is further strengthened by the fact that, no matter which pre-trained model we use, the AUC stays around 0.71 to 0.74, except for the roBERTA model pre-tained on twitter sentiment, which performed the worst. We also found that, just by changing the loss function or slightly increasing the learning rate caused the model to completely fail and never optimize. Thus, due to the sensitivity of this model in terms of the hyper-parameters, next steps might involve thoroughly tuning these hyper-parameters to achieve top accuracy with an optimal model. However, due to MMF (a bimodal architecture made by Facebook) being able to achieve AUC of 0.80 or more on this dataset, we feel that this model is also being restricted by using a CLIP model, in which the model learns to pair images and text, even though the images and text might not be obviously inherent in how they pair up. For example, there is no obvious reason for a model to learn why a photo of a desert is paired with a phrase such as "Look how many people love you", as the image

Pre-Trained-Model	Parameters	Accuracy + AUC
Hate Speech (images with text)	Madgrad, BCELoss	0.62 and 0.73
Hate Speech (images with no text)	same settings	0.62 and 0.74
Hate Speech	weight decay reduced to 0.0002	0.59 and 0.71
Hate Speech	cross-entropy loss	0.50 and 0.50
Hate Speech	lr=7e-4 over 2e-4	0.50 ad 0.45
Meme Sentiment (FinBERT)	Madgrad	0.58 and 0.70
Twitter Sentiment (roBERTa)	Madgrad	0.57 and .67
General BERT uncased	Madgrad	0.58 and 0.73
Hate Speech	Adam optimizer	0.61 and 0.73

Table 1: Published CLIP Model performance.

itself does not necessarily project absence on its own, and could easily be the image paired with any other text related to sparsity or emptiness. Furthermore, many memes use similar texts or images, with slight alterations to the second medium, which might make a model trained on pairing the medium do poorly.

This experimentation also showed us that Adam equally performs well on this data and model, which was carried into our trials with a dual-stream model next.

3.2. Dual-Stream Bimodal Encoder Model

While expected to perform the worst of all our models, we still developed a dual-stream model to compare against the uni-stream model to see if such a model, in which the image and text are encoded separately and streamlined into a single classification model, could pick up some of the interactions between the text and image and produce similar accuracies as the model above.

However, as expected, the model was unable to learn the data, and performed no better than a simple random model, albeit able to drastically over-fit the training data, with training accuracy reaching over 95 percent, and validation accuracy never able to get passed around 55 percent.

The images were encoded with a pre-trained google ViT model from Hugging Face, and the text was encoded with a pre-trained finBERT meme sentiment model from Hugging Face. The image encoder was chosen for its performance in CLIP models and its computational efficiency, while the text encoder was chosen for being trained specifically on meme text sentiment, which we felt would perform better than a general model trained on hateful speech (since the text itself in a meme is not necessarily outright hateful), and better than a general language model, due to the special syntax patterns present in the culture of memes. Encodings were captured for train and validation data, and cached in-memory to avoid RAM over-filling. Encodings and final hidden layers were saved per data point for both vision and text, and concatenated together to create a final tensor which would be passed into a series of linear layers for classification. Adam optimizer and BCELoss were chosen due to performance in previous uni-stream model, and learning rates were fine-tuned. When learning rates were too high, the model was unable to perform, without any improved accuracy in the training error or accuracy. Finally, when learning rate dropped to 0.005 or less, was the model able to quickly learn and over-fit on the training data, despite getting near random-guess accuracy on the validation set. To avoid over-fitting, dropout layers were added between the linear layers, and linear layer size was experimented with, but the model was either unable to learn at all, or over-fit much too quickly, showing that this architecture, at its base form, was unfit for this classification task, which makes sense, given our assumptions about how memes are interpreted.

Due to the image and text being encoded separately, the model is unable to produce meaningful encodings that can be learned to detect hate any better than a model trained on any single one of the mediums, image or text. Upon further research, we found other projects that used only a single medium for prediction, such as this model that just predicted on the meme's text, and such models are also unable to produce anything better than 50-55 percent accuracy, with similarly poor AUCs. Thus, this hinted to us that our previous suspicion was, in fact, correct, and that separate encodings for image and text is simply unable to pick up the intricate interaction between interpretation of the text and photo, thus serving no better than if training on any one of the mediums separately.

3.3. Image Captioning (Image-to-Text) & Text Similarity Model

Below are the accuracies of test set for the Image Captions generated by the Hugging Faces Vision encoder decoder model:
Greedy:True, Epochs:3, Accuracy/AUC: 62.2%
Greedy:True, Epochs:5, Accuracy/AUC: 62.8%

Greedy:False, Epochs:5, Accuracy/AUC: 62.8%
Greedy:True, Epochs:10, Accuracy/AUC: 62.8%

As we are using pre-trained models, it did not make much difference in the way the model is interpreting the generated Image caption for Greedy versus Non-Greedy Captions or when we increased the number of epochs for training. We wanted to try using large BERT uncased model, but we ran into Cuda memory issues. **Figure 3** shows the Confusion Matrix for the validation dataset. As we can see, there are more true negatives. Below are a few examples on the behavior:

Figure 4: Image (a) shows Michelle Obama and compared her to a Gorilla which is a hateful meme where as the image (b) is a Non Hateful meme where they are paying tribute to a Gorilla that died. Both Text Similarity and MMBT models could not differentiate the meaning of the Non Hateful Image. Both models predicted both the images to be hateful.

Figure 5: Image (a) show a group of people protesting which is hateful (b) is a Non Hateful meme where a woman tried to cook something and it is burnt. Both Text Similarity and MMBT models were able to predict these 2 memes correctly. where as both failed to predict the image(c) which is a hateful meme. We think that the models could not identify the importance of the Swastika symbol shown in the image. So, the models need to have more training on all the objects and symbols visible in the images and what they represent.

Figure 6: Image Captions generated by the pre-trained model for image (a) is "a woman and a dog are playing frisbee in the grass" and for image (b) is "a man is talking on a cell phone" and the Text Similarity predicted wrongly for both of these images, thus making the caption unable to provide value to the final prediction, whereas the MMBT model predicted correctly for both of these, perhaps due to the image features.

Thus, we think training these models more on Sentiment analysis, Symbols, Famous People etc will improve the prediction capability of these models.

3.4. Uni-Stream VisualBERT Encoding Model

To execute our own single-stream model, we used a detectron2 model pre-trained via a mask-RCNN on the coco dataset. Then, 5 levels of features are returned via the ResNet model, which are used to create box proposals. Finally, a visual bert model pre-trained on the bert-base-uncased dataset is used to embed the images and text.

This architecture we believed combined the advantages of the previous models into one model, combining object detection, which was part of our goal with image captioning, and image feature extraction, which was the goal of the visual embeddings of the single-stream model. Despite building the entire model however, we were unable to complete training, due to extreme memory constraints which became too expensive and computationally large for our virtual machines. Model size and required memory in general was a great challenge across all models, also affecting our batch sizes usable for training and our model parameter count. Thus, for future steps, we recommend that this model be run on a proper setup, and compared to the other models.

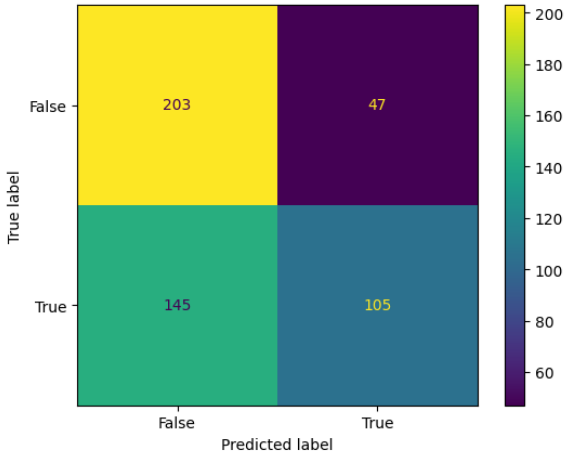
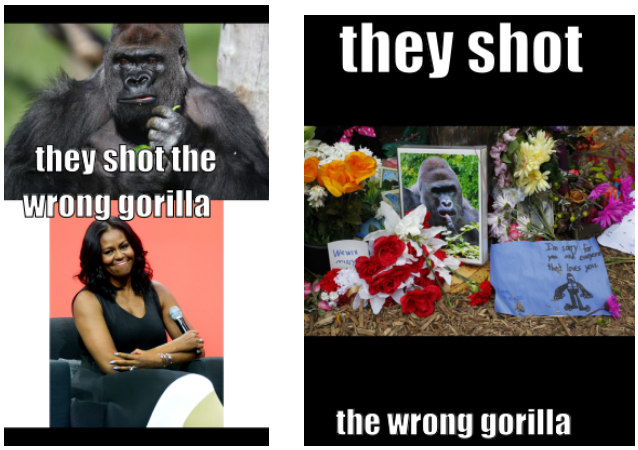


Figure 3: Text Similarity - Confusion Matrix



(a) Hateful (b) Not Hateful

Figure 4: Importance of knowing people

(a) Hateful (b) Not Hateful (c) Hateful

Figure 5: Importance of understanding symbols

(a) Hateful

(b) Not Hateful

Figure 6: Importance of failures of image captioning for memes in which the captioning model is unable to decipher the image content, such as the wheelchair being considered a frisbee

4. Challenges and Future Recommendations

Throughout the development of the project, the main struggle we faced with all models was memory and cost. To run the image masking model, for example, the processing time alone took multiple hours on a high-memory high-CPU, 2 GPU virtual machine, and in the end, it did little to improve accuracy when tested against the raw images. Other challenges we faced were package inconsistencies and deprecation. The top winning models included architectures that are now not runnable with Python 3.8, and to use our virtual machine and bucket memory setup, we were restricted to newer versions of pythons in which the original notebooks' packages were deprecated, even including MMF models.

Furthermore, we strongly feel that better captioning and object detection models could have made major improvements in the performance across all relevant models, to capture aspects of the images that create the common ironic messages of memes. Furthermore, if there were models that were able to detect important figures, religious representations, and on racial slurs, it could also improve the model, as many hateful memes portray prejudice toward religious and ethnic groups, or use a political figure figuratively to represent some prejudice, such as Hitler. However, the hateful message behind some hateful memes are deeply engrained in popularized hateful opinions, which creates an omitted variable bias in modeling, as these cultural references cannot be understood by the model, such as in figure 7.

Finally, each model seems to have its very specific failures across different types of memes. The image-to-text model does well with images with obvious objects, actions, or scenes, but does not do well with figurative representations. The vision-encoding-based models seem to be able

Figure 7: Importance of omitted variable bias and imbalanced data in classifying hateful memes: Without cultural context on popularized hateful opinions, a model would have a very difficult time classifying this, especially given that this hateful message is not common in the dataset

to detect visual tropes that are shared across many hateful memes, such as goats and rainbow flags, but also do poorly with uncommon symbols and in situations in which object detection performs better. Thus, we concur with other winning models that, to achieve top-accuracy, ensembling methods are required to exploit the advantages of each model, while learning in which situations the outputs of each model in the ensemble should be higher-weighted. While we were unable to develop such a method due again to memory constraints, we feel our image-to-text and unistream models could easily be ensembled, and thus perhaps are not poor models, but are simply weak learners on their own.

5. Work Division

Work contributions for all team members is provided in Table 2.

Student Name	Contributed Aspects	Details
Sandeep Arisetti	Implementation and Analysis	1. Trained Image Captioning Model on Flicker Data-set and generated Image Captions. 2. Image to Text - Used nlpconnect/vit-gpt2-image-captioning (VisionEncoderDecoderModel) pre-trained image captioner to generate Image captions to all the images 3. Text Similarity - Used Pytorch transformer finetuning that uses a 'bert-base-cased' BertForSequenceClassification pre trained model that takes Captions generated in the above step and compares it will the Captions on the images. Trained and evaluated the model to detrmine whether the image is a hateful meme or not 4. Text Similarity - Explored to use bert-large-uncased pre-trained model as well, but ran into CUDA memory issues 5. MMBT - Explored CLIP encoder for Multimodal Bit Transformer (created by Rostyslav Neskorozhenyi) which produced accuracy better than the above model
Tyler Lang	Implementation and Analysis	1. Performed the experiments with hyperparameters and pre-trained models with the MMBT notebook (by Rostyslav Neskrorozhenyi). 2. Developed dual-stream model which encodes the images and texts separately, and combines them into one final classification model using Keras. 3. Coded and performed object detection using a pre-trained DETR Image Processor model, and a pre-published photo sentiment analysis model which is trained to detect faces and facial expressions. Both ultimately did not assist in modeling and were then dropped. 4. Ran base model using just the meme text and a roBERTa model to understand the importance of the bimodal data format.
Marc Evans	Implementation and Analysis	1.Used keras OCR to remove letters from images 2. Ran Visual Bert model (hit memory issues), but at the very least produced feature extraction on images at different complexity levels using detectron2, creating visual embeddings and comparing to text embeddings using BERT to model visual reasoning with VisualBert (with the intent to train on the model output). 3.Tried a few different avenues for analysis including: zero-shot object detection using OWL-ViT, sentiment analysis for comparison between image and text to detect sarcasm or congruency, identifying early fusion/late fusion or single stream/dual stream as an important aspect to the project 4.researched methods such as NER as possible avenues for inputs to object detection

Table 2: Contributions of team members.

References

[1] Hateful Memes Dataset
https://www.kaggle.com/datasets/parthplc/facebook-hateful-meme-dataset

[2] Bert Base Uncased Model
https://huggingface.co/bert-base-uncased

[3] Naiver Stokes method
http://elynxsdk.free.fr/ext-docs/Inpainting/todo/inpainting%20with%20the%20Navier-Stokes%20Equations.pdf

[4] Image captioning with visual attention
https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/text/image$_c aptioning.ipynb$

[5] mask-RCNN on the coco dataset
https://github.com/facebookresearch/detectron2/blob/main/configs/COCO-InstanceSegmentation/mask$_r cnn_{R1}01_F PN_3 x.yaml$

[6] detectron2 model
https://github.com/facebookresearch/detectron2

[7] general bert-uncased data
https://huggingface.co/bert-base-uncased

[8] sentiment analysis from twitter data
https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

[9] zero-shot object detector as well
https://huggingface.co/docs/transformers/main/en/tasks/zero$_s hot_o bject_d etection$

[10] Hugging Face's pre-trained DETR Image Processor model
https://huggingface.co/docs/transformers/model$_d oc/detr$

[11] visual bert model pre-trained on the bert-base-uncased dataset
https://huggingface.co/docs/transformers/model$_d oc/visual_b erttransformers.VisualBertForPreTraining$

[12] pre-trained VisualBERT model
https://huggingface.co/docs/transformers/model$_d oc/visual_b ert$

[13] pre-trained google ViT model
https://huggingface.co/google/vit-base-patch16-224

[14] pre-trained finBERT meme sentiment model
https://huggingface.co/jayanta/ProsusAI-finbert-sentiment-finetuned-memes

[15] BertForSequenceClassification model
https://huggingface.co/transformers/v3.0.2/model$_d oc/bert.htmlbert forsequenceclassification$

[16] pre-trained image captioning model
https://michael-franke.github.io/npNLG/08-grounded-LMs/08c-NIC-pretrained.html

[17] DataDriven
https://www.drivendata.org/competitions/70/hateful-memes-phase-2/

[18] Model to predict based on the meme's text
https://www.kaggle.com/code/ashutosh619sudo/facebook-hate-memes-competition-with-roberta