Turning a Meme Into Words: Project Report CS 7643

Caitlin Shener
Georgia Tech
cshener@gatech.edu

Isha Saini
Georgia Tech
isaini6@gatech.edu

Shide Qiu
Georgia Tech
sqiu74@gatech.edu

Abstract

Memes are widely spread on social media platforms and consist of texts and graphics to create humor. Unfortunately, sometimes memes uploaded on the Internet need to be censored to curb hate speech. This poses a new challenge for multimodal classification which requires both visual and linguistic understanding of the meme. We find that using textual summarizations of the memes along with the text embedded in the image can help language models determine if the meme should be considered hate speech.

1. Introduction/Background/Motivation

This work investigates Meta's "Hateful Meme Challenge."[9] The goal in this challenge is to predict if a Meme, or image with text overlaid on top, is hate speech. Meta clearly defines hate speech as:

> "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."

This is a difficult task because while both the text and image of a meme can be harmless on their own, put together they can create hate speech. This subtlety is difficult for a model to detect and makes the problem "multimodal."

Historically, in order to curb the spread of hate speech, malicious content had been inspected by humans. However, this cannot be done at the current scale of Internet. Recently, multimodal classification problems such as memes detection attracted the attention of many machine learning researchers."[12] One approach was Late-fusion. Late-fusion multimodal systems classify memes for image and text seperately before attempting to fuse the results. A Late-fusion system is easy to build but less effective than Early-fusion methods due to its limited understanding of the text and image together. Early-fusion, however, combines different modalities (image and text) first and then classifies the content all at once.[9] Early-fusion is more effective than Late-fusion because its output is based on the entire multimodal content.

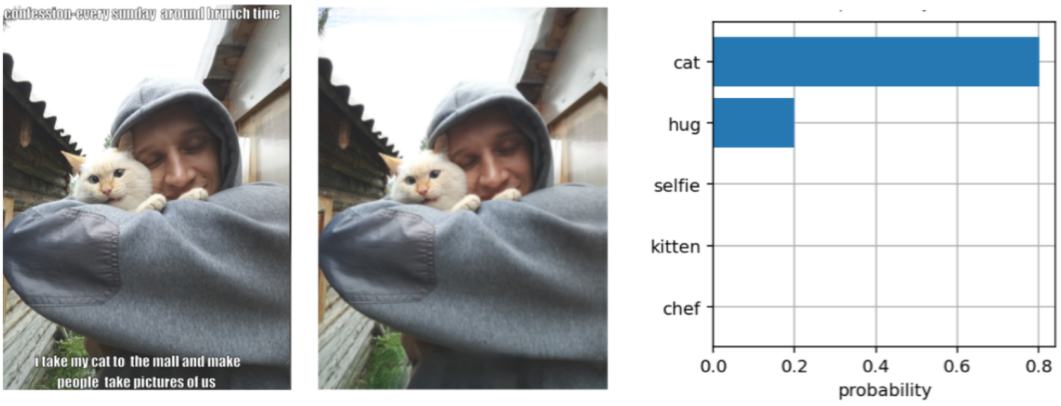
Currently, the winning algorithm to solve this The Hateful Memes Challenge was developed by Zhu.[15] The author combines four different VL transformer architecture ensembles (VL-BERT [11], UNITER [6], VILLA [14], and ERNIE-Vil [13]. The optimal performance reaches 0.845 AUROC. The main limitation of this method is that the pre-trained models applied to hateful memes detection are trained on everyday objects and simple captions which comprises of a vastly different domain than potentially hateful memes.

The motivation behind this project and challenge are two fold. The first motivation is to create a model that does better on multimodal problems. The second motivation is to increase Artificial Intelligent's ability to automatically identify hate speech. This is important because with vast amounts of information on the web, automatic identification is key to being able to limit hate speech and the spread of hateful content.

Meta created, paid for, and provided the data used in this paper as a part of their challenge. [9] Overall, the dataset has 10,000 images labeled as hateful or not. Because the goal is to identify hate speech, many of the images can be considered offensive, insulting, or threatening. Meta created these images synthetically to look like real user created Memes, but sourced images from Getty Images for copy right reasons. Meta then used a human annotation process to label the memes as hateful or not. For every meme the human annotation process found hateful, they also found an alternative image or caption to make the image no longer hateful. This forced well preforming models to be multimodal.

2. Approach

Our approach started with cleaning the images. We created a new dataset of images without the text overlaid in the



Original Meme **Meme Cleaned of Text** **Probability of Tags from Clip**

Figure 1. An example of the process of cleaning and then tagging a MEME. In this example we use 01389.png from the Hateful Memes Challenge. [9]

meme for better image processing as seen in Figure 1. For this, we re-used the code from the winning team [15] of the Hateful Memes Challenge. The code reads the input image and detects the text in the image using bounding boxes, then it generates a text segmentation mask (which essentially is a black image with white mask on the region where text was detected), then the generated mask and image are fused to replace the text and inpainted to fill the missing pixels from original image. An example of this is Figure 1 in which we see a meme before and after cleaning the text.

Our approach focused primarily on text based models. We used BERT[7] pretrained on the English corpus as the base model and incorporated the visual aspect by creating written descriptions of the images. To start with, we trained solely on the text in the meme and nothing from the image. We used text from memes as input and labels (hateful/non-hateful) as targets. We referred to code in "Classification with BERT in Pytorch" [3] as reference and built on top of it. We use the embedding token ([CLS]) from BERT model as input to our classifier, a linear layer followed by ReLU activation which gives us a vector of size 2 with probabilities of hateful vs non-hateful text, we choose the maximum out of the vector and use the corresponding index to get class labels(0 = non-hateful, 1 = hateful).

We tuned the hyperparameters to get the best validation results on this alone. As seen in Figure 2, improvements were very little with the MEME's text only approach. Figure 3 is the confusion matrix for the trained BERT. We can see the mis-classification errors for each of the classes. The highest mis-classified bucket is when the meme is hateful but our model detects it as non-hateful (we have 392 such memes).

We provided some of the examples of miss-classified

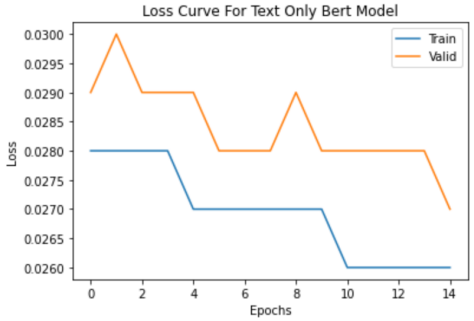


Figure 2. Loss curve from tuning BERT

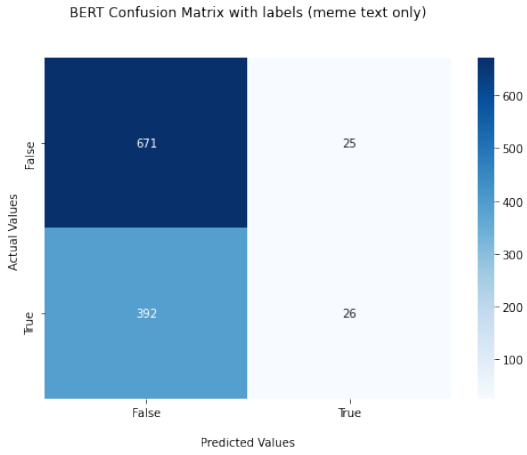


Figure 3. BERT Confusion Matrix

memes in Figures 4 and 5. We removed the text to let the memes illustrate our point without including potential hate speech in our paper. After analyzing these examples we de-

cided to expand our experiments to include features from images in form of text.

Figure 4. An example of a mis-classified hateful meme marked as non-hateful. The text in the uncleaned version implies something hateful about the subject.

Figure 5. Another example of a mis-classified hateful meme marked as non-hateful. The text in the uncleaned version also implies something hateful about the subject.

2.1. Text Paragraph Descriptions

We added additional textual descriptions of the images to see if it would improve the model. Our tagging approach was largely inspired by Zhu(2020)'s approach. One large difference was that Zhu(2020) fed the tags into an ensemble of multimodal models, namely, VL-BERT [11], UNITER [6], ERNIE-Vil [13], and VILLA [14]. We wanted to learn how the model could do without the images and only the tags. To test this, we created more specific and language based tags to incorporate into descriptive summaries about each image. We broke these tags down into two parts – human and entity based. Both of these tags took the form of a sentences.

2.1.1 Human Description Sentences

The first sentence summaries all of the humans in the meme. First we used FairFace [8] to apply a bounding box to each person and tag their race, gender, and age. FairFace is pretrained on YFCC-100M Flickr dataset.[5] Four race groups are defined: White, Indian, Black, and Asian. Male and female are the gender classified by the model. Age groups are : 0-2, 3-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70+. We applied the model to all of the images in the dataset. For images without a face outputs "No face detected". For image with one face, the model applied a bounding box to the face and cropped it to generate a new image with only the face in it. For images with more than one face, the model applied multiple bounding boxes to all faces and cropped them separately to generate new images with seperate faces. In the end, the model tagged the race, gender and age for every face (cropped image).

Then we also used a facial emotion recognition python client [2] to tag the emotion of each face (cropped image) generated by FairFace. The emotion is classified as: angry, disgusted, fearful, happy, neutral, sad and surprised.

We combined all of these tags into one sentence per person using F strings such that a sentence reads "This is an image of a 'age tag' year old 'race tag' 'gender tag', 'emotion tag' ." . Figure 4 is an example of successful classification and the sentence is "This is an image of a 50-59 year old White Male, Happy." Figure 1 is a mis-classified example of this for which the sentence becomes: "This is an image of a 40-49 year old Indian Male, NoEmotion." The FairFace model detected the face location as expected but made an inaccurate prediction for race and gender. As for the emotion model, it failed to detect the face.

2.1.2 Entity Description Sentences

We then also created entity sentences to summarize the entities in the image. We did this by starting with the results from Zhu[15]'s use of Google Vision Web Entity Detection to capture the image's context. We downloaded that paper's annotations[1] to start as the base of our own annotation method. These downloaded entity tags were often along with a lot of noise so we scrapped the annotations for the "webEntities" and "bestGuessLabels" from these files. We put all of these words and phrases into one large list and counted the most commonly occurring ones. From there, we saved the 300 most common words and phrases that

were not prepositions or having to do with photography. For example, many of the tags were 'Getty Images' or 'Stock photography' which would not be helpful to our task. We then used these 300 labels as new classification groups. We feed all 10,000 images into CLIP [10] with the 300 phrases as potential classifications. We chose CLIP for the classification because of its ability to predict a much wider set of visual concepts than other models as our 300 labels are different from typical image classification tasks. For example, one of the labels is "domestic violence" and another is "paintings by adolf hitler.' We saved all the phrases with a softmax cross entropy score of above 5 percent as the new tags for each meme. For example, in Figure 1, we see a meme that is mostly classified as a cat and then a hug. Then we take all the new classifications and turn them into a simple and sometimes grammatically incorrect sentence such as: "There is cat and hug."

Finally, we combine the 'Human Description Sentence', the 'Entity Description Sentence', and the text from the meme to create one short paragraph summarizing the meme. In the example above the descriptive paragraph is:

> "This is an image of a 40-49 year old Indian Male, NoEmotion There is cat and hug. confession- every sunday around brunch time i take my cat to the mall and make people take pictures of us"

This summary reduces down the picture to very simple terms. One down side to this approach is that the summaries might be too simple to properly convey the visual part of the meme. This was a compromise for our team due to the problems we encountered. Namely, our original goal was to use tags as well as the text and image in a multi-model approach, but we found implementing VL-BERT to be consistently problematic with the environments we stood up in Google Cloud Platform. We think that being able to use a model that considers text and images might have gotten us better results, but made the pivot to all text based inputs into BERT to fit within the time restrains.

3. Experiments and Results

We ran following experiments to compare our model performance on meme data:

1. Pre-trained BERT model performance on meme text without hyperparameter tuning.
2. Fine-tuned the pre-trained BERT model and measure performance on meme text.
3. Add more complexity to classifier by adding Linear layers followed by ReLU activation and Batch Norm layers to original architecture.
4. Add entity tags extracted from image to meme text and measure performance of model on new text.
5. Add entity, emotion tags from image and measure model performance on new text.
6. Add entity, emotion, race, gender, age tags extracted from image to meme text and measure model performance on that.

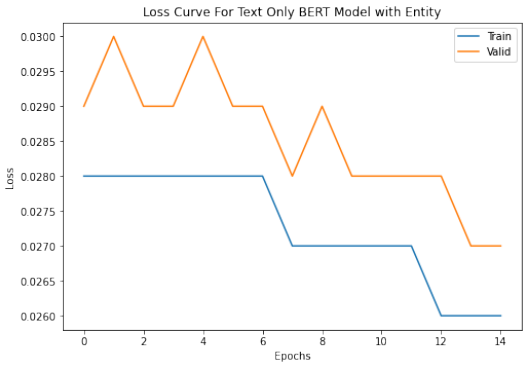


Figure 6. Loss curve with entity

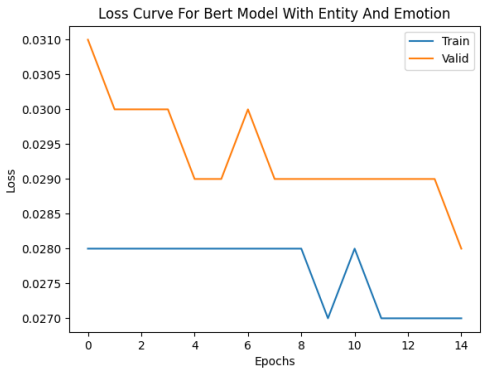


Figure 7. Loss curve with entity and emotion

When we used the pre-trained BERT model as is, the model overfits the data. We noticed this because training loss continued to decrease over epochs while the validation loss began to increase. This told us that the model was overfitting on train data and not able to generalize well on the validation set. The final test accuracy is 0.62 and we set it as our baseline. One reason for the poor performance is the random hyperparameters. To overcome this, we ran a grid-search on 5 hyperparameters. Due to the time limit and a two hour training time per model, we studied the following combinations:

1. batch size (24)
2. epochs (10, 15)

Model	Train acc	Validation acc	Test acc
Pre-trained BERT with no hyperparameter tuning	0.88	0.58	0.62
BERT with hyperparameter tuning	0.66	0.62	0.63
BERT with entity tags	0.65	0.636	0.643
BERT with entity and emotion tags	0.648	0.621	0.633
BERT with entity, emotion, and FairFace tags	0.696	0.662	0.674

Table 1. Results from training BERT.

BERT with meme text model parameters	Value (best)
batch size	24
epoch	15
learning rate	1e-6
decay rate	0.05
dropout	0.5

Table 2. The hyperparameters from the best BERT model using meme text only.

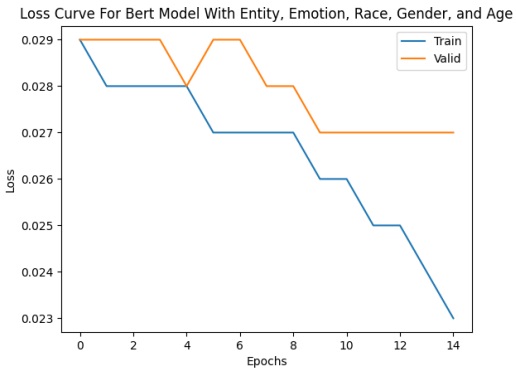


Figure 8. Loss curve with entity, emotion, race, gender and age

3. learning rate (1e-6)

4. regularization parameter (0.01, 0.05, 0.1, 0.5)

5. dropout (24)

We noticed that introducing the regularization (L2 loss) helped reduce the overfitting. We tried with different values of other hyperparameters to find the best model. Our best model configuration is shown in Table 2. We also tuned our classifier architecture by adding more complexity through introducing more linear layers followed by ReLU and Batch Norm layers. The performance from this was not as good as from the simple single layer classifier. The test accuracy is 0.63 which is similar with BERT without hyperparameter tuning. Since BERT only takes the text of memes into account and ignores image, the unimodal model struggles and is hard to succeed.

After finalizing the architecture and running the grid-search, we started looking at the memes that were labeled as non-hateful based on text but were hateful when image and text were combined together. The confusion matrix helped us understand the mis-classification rate.

From the confusion matrix, we observed that adding image features to our text might help our model detect hateful memes better.

We started with adding the entity tags extracted from the meme image. The loss curve is shown in Figure 6. The test accuracy reaches 0.643. The entity tags introduce image information to the model and the late-fusion model boost the performance based on the understanding of both image and text.

Since entity tags only describe the object itself, we added emotion tags to include more information from the meme image. The loss curve is shown in Figure 7. The test accuracy is 0.633 which is lower than BERT with entity tags model. The reason could be some images have multiple faces with different emotions and the model got confused with such data.

The attack on people based on race and gender is very common in memes. Thus, we added FairFace (race, gender, age) tags from the meme to extract these information. As a result, the test accuracy is 0.674 which is the optimal one among all models we trained. The main reason is that it described the image in more details so that it could have a better understanding of both image and text. The loss curve in plotted in Figure 8. The performance details are shown in Table 3.

4. Infrastructure set-up

For any deep-learning project, having a strong infrastructure is really important. We faced multiple challenges initially using Google Collab as it had very limited GPU support and often timed out. Hence we decided to move away from Google Collab and leveraged the following infrastructure (athough we were still limited by the credits available to

us for GCP): 1. GCP vm instance with GPU - 1 x NVIDIA Tesla A100 2. GCP buckets to store results from model training and evaluation. 3. Jupyter notebook set-up on GCP vm

5. Future Work

For future work, we could take our project work further by using state of art models like CLIP and VL-BERT that can learn the context from both image and text together. A model like this might make better predictions than our text summaries. Another interesting area where we could expand on would be our current architecture (i.e. using BERT for text, CLIP from entity extraction and Fairface for other tags) is to study how meme images with multiple faces each having different emotion, race, etc, are learned by the model and evaluate model performance on only subsets. It would be interesting to understand which feature influences the model's decision making. Moreover, the recent success of self-training shows it works better than pre-training in some areas [4]. Self-training as an extra method to utilize additional data (labeled or non-labeled) on the same setup could lead to better performance and it is worth to explore more.

6. Conclusion

In this project, we explored Facebook's Hateful Memes Challenge in detail. We used the BERT model as a baseline. We extracted entity, emotion, race, gender and age tags from the meme image and fused them with the meme's text. The optimal model was found by tuning hyperparameters and the test accuracy reached 0.674. This worked surprisingly well for a text focused approach. Especially when we compare this result to the challenge's introduction paper in which the authors test more state of the art models [9]. Their highest model which was multimodal with pretraining achieved an accuracy of 69.47±2.06. Our model was able to do about as well as all of the other models the authors tried. In the end, we also proposed some possible future work for further improvement to our approach.

7. Work Division

Table 3 outlines each team members' contributions.

References

[1] Data preprocessing. https://github.com/HimariO/HatefulMemesChallenge/blob/main/data_utils/README.md. 3

[2] facial-emotion-recognition 0.3.4. https://pypi.org/project/facial-emotion-recognition/. 3

[3] Text classification with bert in pytorch. https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f. 2

[4] Tsung-Yi Lin Yin Cui Hanxiao Liu-Ekin D. Cubuk Quoc V. Le Barret Zoph, Golnaz Ghiasi. Rethinking pre-training and self-training. 2020. https://arxiv.org/abs/2006.06882.pdf. 6

[5] Gerald Friedland Benjamin Elizalde Karl Ni-Douglas Poland Damian Borth Li-Jia Li Bart Thomee, David A. Shamma. Yfcc100m: The new data in multimedia research. 2015. https://arxiv.org/abs/1503.01817.pdf. 3

[6] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: learning universal image-text representations. *CoRR*, abs/1909.11740, 2019. 1, 3

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. 2

[8] Kimmo Kärkkäinen and Jungseock Joo. Fairface: Face attribute dataset for balanced race, gender, and age. *CoRR*, abs/1908.04913, 2019. 3

[9] Douwe Kiela, Aravind Mohan Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Pratik Ringshia Amanpreet Singh, and Davide Testuggine. The hateful memes challenge: Detecting hate speech in multimodal memes. 2021. https://arxiv.org/pdf/2005.04790.pdf. 1, 2, 6

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 4

[11] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-BERT: pre-training of generic visual-linguistic representations. *CoRR*, abs/1908.08530, 2019. 1, 3

[12] Muhammad Numan Khan Jawad Khan Young-Koo Lee Tariq Habib Afridi, Aftab Alam. A multimodal memes classification: A survey and open research issues. 2020. https://arxiv.org/abs/2009.08395.pdf. 1

[13] Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-vil: Knowledge enhanced vision-language representations through scene graph. *CoRR*, abs/2006.16934, 2020. 1, 3

[14] Linjie Li Chen Zhu Yu Cheng-Jingjing Liu Zhe Gan, Yen-Chun Chen. Large-scale adversarial training for vision-and-language representation learning. 2020. https://arxiv.org/abs/2006.06195.pdf. 1, 3

[15] Ron Zhu. Enhance multimodal transformer with external label and in-domain pretrain: Hateful meme challenge winning solution. 2020. https://arxiv.org/abs/2012.08290.pdf. 1, 2, 3

Student Name	Contributed Aspects	Details
Caitlin Shener	Entity Tags	Implemented entity tagging classification approach using CLIP.
Isha Saini	Image Pre-processing and BERT	Implemented text-based BERT model for hateful text detection by fine-tuning it and cleaned the images for tagging
Shide Qiu	FairFace and Emotion Tags	Implemented race, gender, age, and emotion tagging classification approach using FairFace and facial-emotion-recognition

Table 3. Contributions of team members.