Hate-full Memes Identification

Amirhesam Abedsoltan
University of Southern California
abedsolt@usc.edu

Abstract

This document contains our results and analysis of our models for the "Hateful-Memes" competition by Facebook. The goal was to identify hateful memes, which is a binary classification problem. We first tried to improve the VisualBert model using optimization methods. VisualBert model was originally introduced (Li et al., 2019) and was used for this task (Douwe Kiela, 2020). Then, we introduced our own multi-layer Neural Network and tried to compare the results between this model and the VisualBert model. It turned out the multi-layer Neural Network is performing almost as well as the VisualBert model. It is also faster to train. Lastly, we found out that these models' poor performance on the Development test is due to the distribution shift between the training data set and the development data set. Thus, we tried to do robust learning, which slightly improved the results but still not as good as human performance(63.5% Vs. 84.5%). Finally, as a part of the extra mile, we tried the clustering method using VisualBert outputs. The accuracy increased to 65%.

1 Introduction

Many memes are being uploaded on social media, and not all of them are appropriate or legal. This project aims to have an effective AI algorithm to identify inappropriate memes and remove them from social media automatically. This task, even for a human, might not be easy. However, since the true meaning usually depends on prior knowledge about a sentence or an image, this could be even more challenging for a machine. Also, usually by only looking at one source, meaning image or sentence, one cannot judge the true meaning of the meme, and it's crucial to understand the relation between image and the sentence. This is again challenging for a machine to learn the connection between an image and a sentence. For instance, look at the memes in Figure 1and Figure2. These examples have been borrowed from (Douwe Kiela, 2020).

Figure 1: How picture can change the interpretation of a sentence.

Figure 2: How sentence can change the interpretation of an image.

2 Problem Description

This is a classical 0-1 classification task, and the final goal of this project is to identify hateful memes with acceptable accuracy and ROC. This project is an example of a multimodal learning problem. Our data is naturally image and text. We need to

know the state-of-art tools to extract meaningful information from each source separately and find mutual information between the sources.

3 Data set

Facebook has provided two data sets. One for training, which contains 10000 memes, and another one as a development data set containing 500 memes. We have access to images through their ".jpeg" files, and also, we have access to true labels and sentences through two separate ".json" files for training and dev data set. The data set is publicly available and could be downloaded from the "Drivendata" website.

4 Approaches took

In the figure3 you can see the baselines' results from the original paper (Douwe Kiela, 2020).

Type	Model	Validation Acc.	Validation AUROC
	Human	-	-
Unimodal	Image-Grid	52.73	58.79
	Image-Region	52.66	57.98
	Text BERT	58.26	64.65
Multimodal (Unimodal Pretraining)	Late Fusion	61.53	65.97
	Concat BERT	58.60	65.25
	MMBT-Grid	58.20	68.57
	MMBT-Region	58.73	71.03
	ViLBERT	62.20	71.13
	Visual BERT	62.10	70.60

Figure 3: VisualBert model(from original paper).

We first studied both Vilbert and Vimodelsert models. The VisualBert model is much simpler. Since the baseline results are not that different for these two models, we decided to build upon the VisualBert model and improve its performance. In the next sections, we have discussed the details of our analysis and results. The paper has been organized as below: In section 5, we first discuss the original Visualbert model, and then we try to improve the result using optimization techniques. In section 6, we introduce our Multi-layer Neural Network and compare it to the Visualbert model. In section 7, we try to train several weak learners on different parts of the training data set. Then combine them using Adaboost to get a stronger classifier. In section 8, we do robust learning using our multi-layer neural network. In section 9(extra mile), we use the VisualBert model and clustering to boost accuracy.

5 VisualBert model

VisualBert model was originally introduced in (Li et al., 2019). It is based on Bert's idea (Devlin et al., 2019) which itself is based on attention architecture introduced in (Vaswani et al., 2017). The general idea is that we assign an embedding to each word and image as inputs. After concatenating them, we feed them into a model similar to Bert architecture, basically 12 layers of attention model, plus a sigmoid layer as a final layer for binary classification, See Figure4.

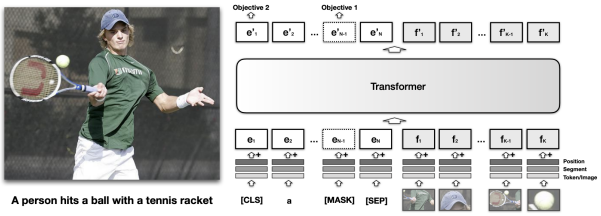

Figure 4: VisualBert model(from original paper).

Also, for input embedding, we used Bert pretrained model for words and Facebook detectron for images (Girshick et al., 2018). Thus, we have assigned a vector of length 768 to each word. Also, we limited the sentences in each meme by 128. Thus for each meme, we have a word feature of the shape of 128 by 768. On the other hand, for each picture, we have a feature of 100 by 768. This was generated using Facebook detectron. Facebook detectron detects 100 objects in each picture and assigns a vector of size 2048 to each object. We then project this vector using a linear layer to space of size 768 to match the Bert output's dimension. Also, note that if there are less than 100 objects in a picture, Facebook detectron uses a default embedding. In the end, we have a feature matrix of shape $(128 + 100) \cdot 768$ for each meme.
Using attention architecture seems really promising because what it basically does is doing dot products between all pair of embeddings and wights them. Thus, the model has great potential to learn the connection between each word and each picture or each pair of words or images.

5.1 preliminary results

We first re-implement the VisualBert model based on the Facebook MMF framework (Singh et al., 2020). We had two reasons for doing this. First, we wanted to try our own ideas and build upon the original model. Secondly, we wanted to see what

the loss value is for the training data set. Because we were suspicious that they might have overfitted the model on the training data set. More precisely, we wanted to see how bad they are overfitting. The results are as below:

Data set	Accuracy
Training	100%
Dev	62%

Table 1: Visualbert results

As we suspected, the loss for training was almost zero, and accuracy was about 100%. Thus, obviously, they are over-fitting to the training data set. Interestingly, in the Facebook MMF code, they do not use any standard optimization method to prevent this over-fitting. We tried some of them like l_2 regularizer. However, it did not really change the results. In the next subsection, we use another technique to prevent this huge over-fitting.

5.2 Prevent over-fitting

We noticed that by using only words as input, we could achieve an accuracy of 60%. These observations suggested that there is a possibility that we are over-fitting to word features. To make sure this is not the case or if this is the case to prevent it, we added two terms to our loss function. Basically, the original loss function, namely L_h, was a cross-entropy loss for binary classification. We also added L_i and L_w, which is a cross-entropy loss for image and word classification tasks respectively, see figure 5.The newly added loss terms do not let the model over-fit on specific features, like words, and prevent losing information on the other feature, like image. So, basically, if we could retrieve the input labels from the outputs of the transformer, which has 12 layers, we still have enough information about both images and words. To make it clearer, note that we assigned an embedding to each object and word in the input; after feeding it to the transformer, we get an output of the same size of $(128 + 100) \cdot 768$, which corresponds to the same input. If we could use the transformer's corresponding outputs to retrieve the original label, we still have enough information about images and words.

the results after fine tuning on α and β is as table 2. As you can see, we have a really high accuracy on images and word classification tasks, which means we are not losing information. Therefore,

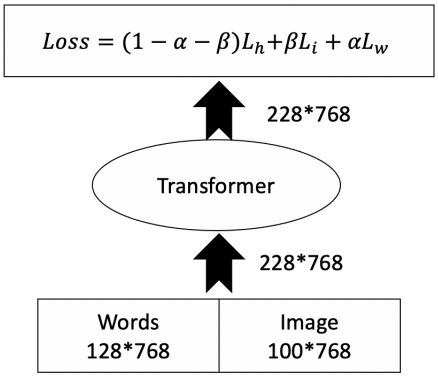

Figure 5: adding additional loss terms to prevent over-fitting.

we are not over-fitting on a specific feature. However, the accuracy of the hateful task is still low and around 62%. Thus, we concluded that over-fitting is not the main reason we have a poor performance on the development data set.

Task	Accuracy
image	98%
words	98%
Hateful	61.5%

Table 2: Visualbert results modified loss funtion

6 Multi-layer Neural Network

From the beginning of the project, we were not sure about the superiority of the VisualBert model. Although the VisualBert model seemed very promising, we wanted to see the performance of a much simpler model. Thus, we decided to use a sequential Neural Network model and compare its results to the Visualbert model. Also, instead of using Facebook detectron to generate features from our images, we decided to use Resnet-152 to generate features. To do this, we used the last layer of the pre-trained Resnet-152 model with dimension $2044 \cdot 7 \cdot 7$, and using a series of linear layers, converted it to $20 \cdot 768$ to match Bert's outputs dimension. Also, for faster training, we limited our sentence to 64 words instead of 128 words earlier. Thus, input would be of size $(64 + 20) \cdot 768$. You can see the final model in Figure 6.

Results using this model is as table 3. As you can see, it's performing comparably to the VisualBert model with the additional advantage that it trains much faster due to its simplicity and a smaller number of parameters.

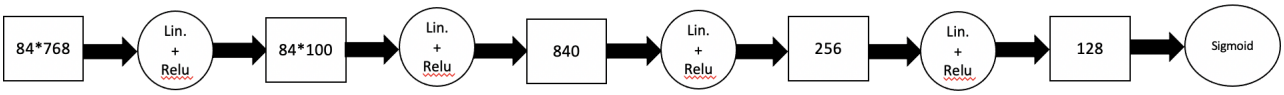

Figure 6: Multi-layer Neural Network model.

Data set	Accuracy
Train	100%
Dev	62%

Table 3: Multi-layer Neural Network results

7 Adaboost

The idea of training several copies of a model on different data sets and creating so-called weak learners has been around for a while (Freund and Schapire, 1999). Also, the boosting method was shown very useful in the winner solution of the Netflix competition (Koren, 2009). Inspired by these ideas, we wanted to try the Ada-boost method to boost our accuracy for this project. We divided the training data set into 5 equal parts, each with a 2000 data. Then, we added 500 data from other parts to each part to create some non-empty intersections between training data sets of different weak learners. To sum up, we trained 5 Neural Network models, which were introduced in the previous section, on 2500 data. Then, using the well-known Ada-boost method, details of this well-known method can be found here (Freund and Schapire, 1999), We create a model that uses a combination of these 5 models for prediction. The final results were as table 4.

Data set	Accuracy
Train	100%
Dev	58%

Table 4: Multi-layer Neural Network results

As you can see, the results of the development data set decreased. The reason is that 2500 data was not enough for the models to be trained on, and each model is hugely over-fitting on its own training data set. Thus, due to the lack of enough data, we did not continue this approach.

8 Distribution Shift

After trying these ideas and realized that standard approaches could not resolve the over-fitting, we concluded that there must be a deeper reason for this poor performance on the development data set. One possibility that we thought of was the idea of distribution shift. The details about this concept can be found here (Ovadia et al., 2019). However, distribution shift happens when you train your model on some data set with a true distribution like P but your development data set has true distribution like q and P and q are not exactly alike. The general idea of using machine learning methods is to have similar training, development, and test data sets. However, this is not always the case. There are different situations in practice that gradually, data distribution starts to move from the original distribution. In this project, we wanted to check if there was a shift in distribution between the training and development data set. Thus, we divided the training data set into a new development data set with 500 data and a new training data set with 9500 data. We used our Neural Network model again. you can see the results in table 5:

Data set	Accuracy
Train	100%
Dev	82%

Table 5: Multi-layer Neural Network results on divided taring data set

As you can see, the accuracy shifted to 82%, which is really high compared to human performance(84%). This new result showed that our conjecture was right, and there exists a distribution shift between the training and development data set. One way to remedy this problem is by using the ”Robust learning” method. In the next subsection, we introduced one of these methods and tried it on our data set.

8.1 Robust Learning

Although there are many works on ”Robust learning”, a well-known and general framework for this problem does not exist yet. To make the idea of ”Robust learning” clear, we borrow an example from (Goodfellow et al., 2015). In this paper,

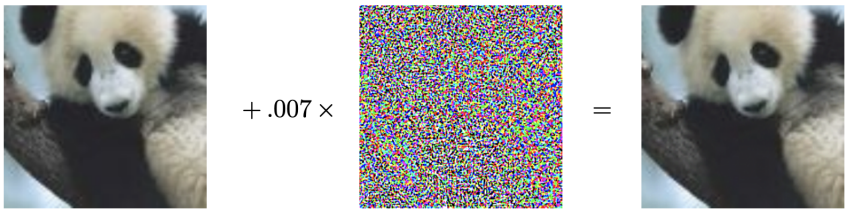

Figure 7: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014) on ImageNet.

the idea of "Robust learning" has been explained for adversarial attack problem. Look at Figure 7; as you can see, the model confidently classifies the picture of panda as a gibbon by only changing the input intangibly. The reason for this the phenomenon can be seen in the Figure 8.

the

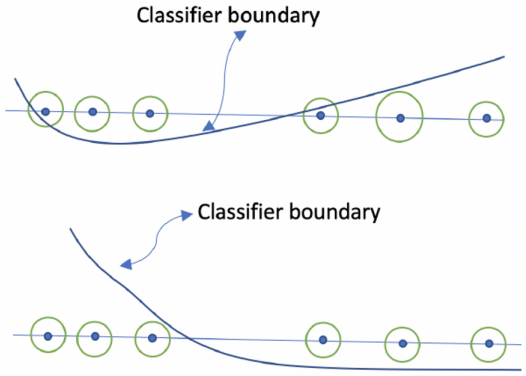

Figure 8: Robust Vs. Non-Robust classifier.

In figure 8, assume each point is a sample in training set in the high dimensional space. Usually, we consider a latent feature in the last layers of the neural network. Also, assume the figures' boundary is between class "panda" and class "gibbon." As you can see, in the classifier on the top, if we change the input features intangibly, we would cross the boundary and miss-classify the picture. On the other hand, in the second classifier, if we change the features around a ball with radius ϵ, we would not cross the boundary, and we would predict the picture correctly. There are many ways to formulate this idea and develop algorithms to implement. Inspired by (Carlini and Wagner, 2017) we formulate this idea as a MinMax optimization problem. Assume $x_1, x_2, ..., x_N$ are each the latent features of the second last layer of neural network in Figure 6. Each x_i represents a sample in the training set. To do a robust learning we modify the cross-entropy loss to new loss function L_{new} defined below:

$$min_w L_{new} = \sum_{i=1}^{N} max_{\delta_i, \text{s.t}|\delta_i| \leq \epsilon} l(x_i + \delta_i; w, y_i) \tag{1}$$

Where l is the cross-entropy loss; in other words, we first define an ϵ, which is the allowed radius to move around a sample. Then, for each sample, we define a δ, which shows how far we move from that sample. We are looking for model parameters, namely W, that minimize the worst case of δ. To formulate the worst case of δ, we use the loss function's maximization for δ. To sum up, we chose the model parameters that minimize the worst case of cross-entropy loss for choices of δs. This optimization problem is not convex. Thus, finding the global minimum is challenging. However, we used ordinary Stochastic Gradient Descent(SGD), and to find the gradient of L_{new} we used well-known Danskin's theorem (Al-Dujaili et al., 2018). The results are shown in table 6.

Data set	Accuracy
Train	100%
Dev	63.5%

Table 6: Robust learning results.

As you can see, the result improved slightly, but it is still far behind human performance. However, this is promising, and for future works, we will try to use more sophisticated robust learning methods to improve the results even more.

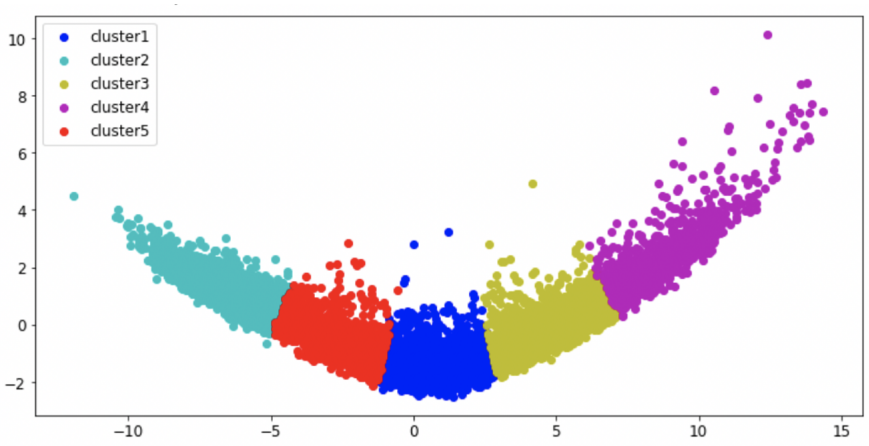

Figure 9: data projection on first two PCs and their clusters. (k=5)

9 Clustering(Extra Mile)

From the beginning of the project, we were interested in the clustering idea. The intuition behind clustering for identifying hateful memes seemed very appealing because it makes sense that in high-dimensional space, the hateful memes be closer to each other or, in other words, be in the same ”cluster.” To test this simple intuition, we used the last layer of the VisulaBert model's features to represent each meme in high-dimensional space. We used the k-nearest neighbor algorithm with k=5 and then used (Principle Component Analysis)PCA to visualize the results. Note that more than 70% of the total variance was captured in the first 2 PCs. Thus, we project the training data set on the first two PCs. You can see the results in Figure 9. Also, below in Figure 10 you can see the percentage of Hateful and non-hateful of each cluster for both training and validation data set.

Result on Training set:
cluster1: non-hateful: 0.9287821323219553 hateful: 0.0712178676780447 samples:(2373)
cluster2: non-hateful: 0.0491006319883325 hateful: 0.9508993680116675 samples:(2057)
cluster3: non-hateful: 0.9913090711569799 hateful: 0.0086909288430201 samples:(1841)
cluster4: non-hateful: 0.9961389961389961 hateful: 0.0038610038610038 samples:(777)
cluster5: non-hateful: 0.3756906077348066 hateful: 0.6243093922651934 samples:(1448)

Result on Validation set:
cluster1: non-hateful: 0.5126903553299492 hateful: 0.4873096446700508
cluster2: non-hateful: 0.16 hateful: 0.84
cluster3: non-hateful: 0.7831325301204819 hateful: 0.2168674698795181
cluster4: non-hateful: 0.8571428571428571 hateful: 0.1428571428571429
cluster5: non-hateful: 0.3442622950819672 hateful: 0.6557377049180328

Figure 10: Percentage of true labels in each cluster for training and validation data set(k=5).

As you can see, there is an interesting pattern in most of the clusters. In training, except for the 5^{th} cluster, all other clusters dominantly belong to either hateful or non-hateful category. Also, for the validation data set, the result is still valid for clusters 2, 3, and 4.however, it is not really accurate for clusters 1 and 5. The final results as numbers are shown in table 7

Data set	Accuracy
Train	90%
Dev	65%

Table 7: Clustering results.

As you can see, it performs better than VisulaBdert and multi-layer Neural Network models. Interestingly if we increase the number of clusters to 10 and again visualize it using PCA analysis, see Figure 1, you can see that 5^{th} cluster starts to get divided into smaller clusters. Again you can see the percentage of Hateful and non-hateful of each cluster for both training and validation data set in Figure12.

Result on Training set:
cluster1: non-hateful: 0.9120943952802360 hateful: 0.087905604719764 samples:1695
cluster2: non-hateful: 0.995867768595041 hateful: 0.004132231404958 samples:726
cluster3: non-hateful: 0.1 hateful: 0.9 samples:840
cluster4: non-hateful: 0.993326978074356 hateful: 0.0066730219256434 samples:1049
cluster5: non-hateful: 0.050406504065040 hateful: 0.9495934959349593 samples:615
cluster6: non-hateful: 0.991404011461318 hateful: 0.0085959885386819 samples:349
cluster7: non-hateful: 0.029051987767584 hateful: 0.9709480122324159 samples:654
cluster8: non-hateful: 0.450508788159111 hateful: 0.5494912118408881 samples:1081
cluster9: non-hateful: 0.984429065743944 hateful: 0.0155709342560553 samples:1156
cluster10: non-hateful: 0.096676737160120 hateful: 0.9033232628398792 samples:331

Result on Validation set:
cluster1: non-hateful: 0.47651006711409394 hateful: 0.5234899328859061 samples:149
cluster2: non-hateful: 0.8275862068965517 hateful: 0.1724137931034483 samples:29
cluster3: non-hateful: 0.12903225806451613 hateful: 0.8709677419354839 samples:31
cluster4: non-hateful: 0.8085106382978723 hateful: 0.19148936170212771 samples:47
cluster5: non-hateful: 0.23529411764705882 hateful: 0.7647058823529411 samples:17
cluster6: non-hateful: 0.8333333333333334 hateful: 0.16666666666666663 samples:6
cluster7: non-hateful: 0.21428571428571427 hateful: 0.7857142857142857 samples:14

cluster8: non-hateful: 0.375 hateful: 0.625 samples:104
cluster9: non-hateful: 0.694 hateful: 0.30666 samples:75
cluster10: non-hateful: 0.0 hateful: 1.0 samples:8

Figure 12: Percentage of true labels in each cluster for training and validation data set(k=10).

However, the final accuracy on validation is 66%, which is better than before, but it is not a significant improvement.

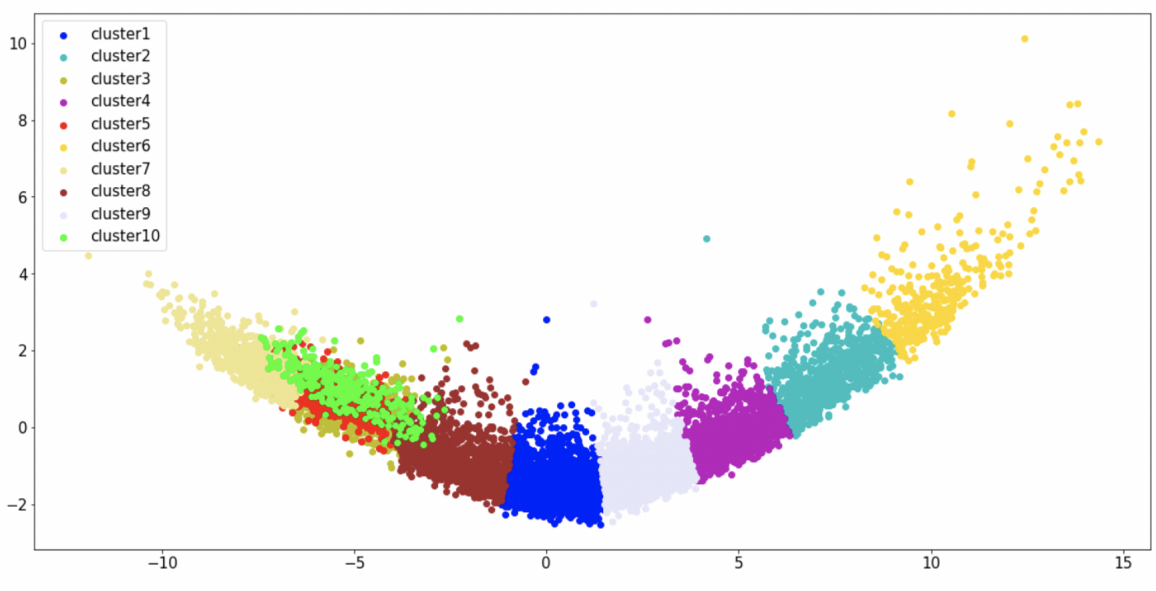

Figure 11: data projection on first two PCs and their clusters(k=10).

10 Related Work

There are many works on hate speech detection in both network science (Ribeiro et al., 2018) and Natural Language Processing community (Waseem et al., 2017), (Schmidt and Wiegand, 2017). Also, there exist several text-only hate speech data sets, mostly based on Twitter (Waseem, 2016), (Waseem and Hovy, 2016),(Davidson et al., 2017). Hate speech detection turned out to be difficult due to unwanted bias (Dixon et al., 2018), (Sap et al., 2019) or different terminologies for hateful definition (Waseem et al., 2017).

On the other hand, there has been little work in multimodal hate speech. Only a few of them consider both image and text. (Yang et al., 2019) Reports that including image features in addition to text immediately boosts the performance in hate speech detection. (Hosseinmardi et al., 2015) Created a data set from Instagram images and their associated comments, which they then label with Crowdflower workers' help. They asked workers two questions: 1) does the example constitute cyberaggression; and 2) does it constitute cyberbullying. Where the former is defined as "using digital media to harm another person intentionally" and the latter is a subset of cyber-aggression, defined as "intentionally aggressive behavior that is repeatedly carried out in an online context against a person who cannot easily defend him or herself" [31]. They show that including the image features improves classification performance. The dataset consisted of 998 examples, of which 90% was found to have high confidence ratings, of which 52% was classified as bullying.

(Singh et al., 2017) Conduct a detailed study, using the same data set, of the types of features that matter for cyber-bullying detection in this task. Similarly, (Zhong et al., 2016) collected a dataset of Instagram posts and comments, consisting of 3000 examples. They asked Mechanical Turk workers two questions: 1) do the comments include any bullying; and 2) if so, is the bullying due to the content of the image. 560 examples were found to be bullying. They experiment with different kinds of features and simple classifiers for automatically detecting whether something constitutes bullying.

11 Conclusion and Future works

We tried different models to identify hateful memes with high accuracy. We started with the VisualBert model, which was one of the baselines introduced in the original paper (Douwe Kiela, 2020). We tried to prevent the VisualBert model from overfitting. However, none of the optimization methods could resolve huge overfitting existed. Then we realized that this overfitting is not because of the model. In fact, it was because of the distribution shift between the training data set and the validation data set. Thus, we needed to do robust learning. Then, we showed that Robust learning actually increased accuracy. However, this improvement was not that huge. For future works, we can use more sophis-

ticated "Robust Learning" techniques. Also, we showed how the clustering idea seems promising and can boost accuracy. Due to lack of time, our analysis for clustering and Robust Learning was limited. Below you can see all results in one table 8.

Model	Validation Accuracy
Visualbert	62%
Multi-layer NN	62%
Ada-boost	58%
Robust learning	63.5%
Clustering	65%

Table 8: Final results.

References

Abdullah Al-Dujaili, Shashank Srikant, Erik Hemberg, and Una-May O'Reilly. 2018. On the application of danskin's theorem to derivative-free minimax optimization.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification.

Aravind Mohan Vedanuj Goswami Amanpreet Singh Pratik Ringshia Davide Testuggine Douwe Kiela, Hamed Firooz. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes.

Yoav Freund and Robert E. Schapire. 1999. A short introduction to boosting.

Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. 2018. Detectron. https://github.com/facebookresearch/detectron.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples.

Homa Hosseinmardi, Sabrina Arredondo Mattson, Rahat Ibn Rafiq, Richard Han, Qin Lv, and Shivakant Mishra. 2015. Detection of cyberbullying incidents on the instagram social network.

Yehuda Koren. 2009. The bellkor solution to the netflix grand prize.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift.

M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. Almeida, and W. Meira Jr. 2018. Characterizing and detecting hateful users on twitter. in twelfth international aaai conference on web and social media.

Maarten Sap, Dallas Card, Saadia Gabriela, Yejin Choi, and Noah A. Smith. 2019. The risk of racial bias in hate speech detection.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing.

Amanpreet Singh, Vedanuj Goswami, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. 2020. Mmf: A multimodal framework for vision and language research. https://github.com/facebookresearch/mmf.

Vivek K. Singh, Souvick Ghosh, and Christin Jose. 2017. Toward multimodal cyberbullying detection.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Z. Waseem, T. Davidson, and D. Warmsley. 2017. Understanding abuse: A typology of abusive language detection subtasks.

Z. Waseem and D. Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter.a.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter.

Fan Yang, Xiaochang Peng, Gargi Ghosh, Reshef Shilon, Hao Ma, Eider Moore, , and Goran Predovic. 2019. Exploring deep multimodal fusion of text and photo for hate speech classification.

Haoti Zhong, Hao Li, Anna Squicciarini, Sarah Rajtmajer, Christopher Griffin, David Miller, and Cornelia Caragea. 2016. Content-driven detection of cyberbullying on the instagram social network.