Meme Classification

Ravi Pushkar (2018MT60790)
Zuhaib Ul Zamann (2018MT60798)
Instructor: Prof. Niladri Chatterjee

Course: MTL782(Data Mining)

1. Abstract

In this article, we are going to describe the models we used for classification of meme given an image and its text. The Oxford English Dictionary defines "meme" as "an image, video, piece of text, typically humorous in nature, that is copied and spread rapidly by internet users, often with slight variations." The task here is to classify an image with its text into a troll(an offensive meme) or a hilarious meme or none. The data-set used for this classification problem can be found here. We are going to describe how models perform using only image or using only text and how they perform when both(image as well as texts) are used. We are going to describe the transfer learning methods we used for classification to achieve better accuracy as the data-set was small to train from scratch. We will describe how the texts were preprocessed and how the text embedding was learned and how different embedding performed.

2. DataSet Description

The data set consists of 2000 images in train folder with a corresponding csv file that describes the text content(if any) of the images. In case the text was missing, $[\#Name]$ is taken as the text of the image. The csv file also contains the image id of the image, the label num, which is the label of image in numerical form(0 for none, 1 for hilarious and 2 for troll), and the label in text form(i,e troll, none or hilarious) as well.
There is also an unlabelled set of 600 images on which the prediction has to be done. Corresponding to those images is a csv file (train.csv), which contains image id and text of the image. But it does not contain the label and label num columns
The dataset contains 5 files

- **train.csv** - the training set
- **test.csv** - the test set
- **sampleSubmission.csv** - A csv file containing an example submission that has to made(i.e. Gives information of the format in which we have to submit).
- **train_images** - a directory containing a subdirectory train_images which contains the training images
- **test_images** - A directory containing test images on which the prediction is to made.

.

3. Text Pre-Processing And Exploratory Data Analysis

The text in the csv files contained a lot of punctuation marks like @,!, etc. Also some of the texts were in uppercase letters while some text was in lowercase letters. Moreover, there were some unnecessary hyperlinks and tags that needed to be removed. We removed all such punctuation marks and converted the text in lower cases. Also, we used regex to remove hyperlinks from the text. Also some of the tags were removed by removing word having word "meme" in them e.g. memecenter, etc. There were many images in dataset with incorrect annotation(text not correct). Some images which were clearly offensive were labelled as hilarious and vice-versa. We tokenized the processed text using word_tokenize method of NLTK library. We removed stopwords like are, the, a etc from the text. The rest tokens were then converted back to string representation by adding space in between two tokens. We learnt embedding on this text.

The dataset was balanced in terms of classes. The number of each of the labels were almost equal. We also plotted the distribution plot of length of tokens in sentences. From the plot most of the tokens had length $\leq$64. We took 64 as the maximum length of the embedding we wanted to learn. There are no repetitions of an image in the dataset and no missing values in the dataset.

There is an image which could not be read using cv2.imread. The image was dropped during training regarding the image as corrupted. We decided to process images at the time of training as images take up a lot of space in kernel and would lead to kernel getting restarted if the processing on images was done all at a time. I images we used image augmentation techniques using Im-

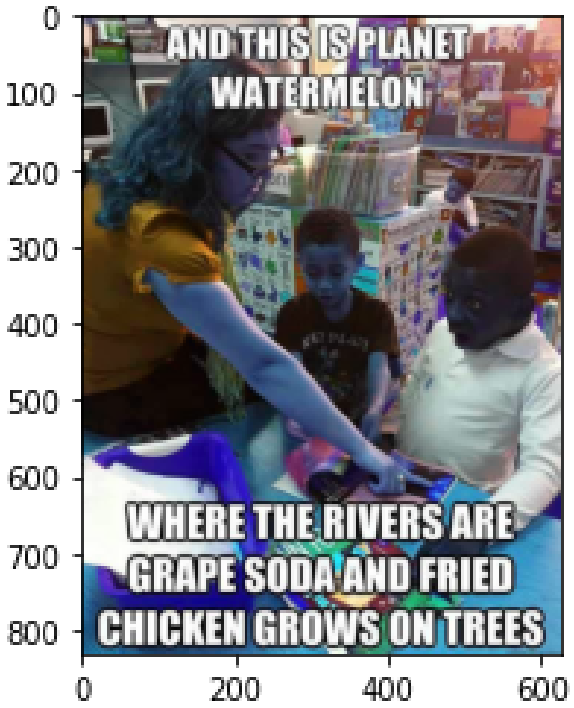

Figure 1: Wrong text annotation: This image was labelled as having no text

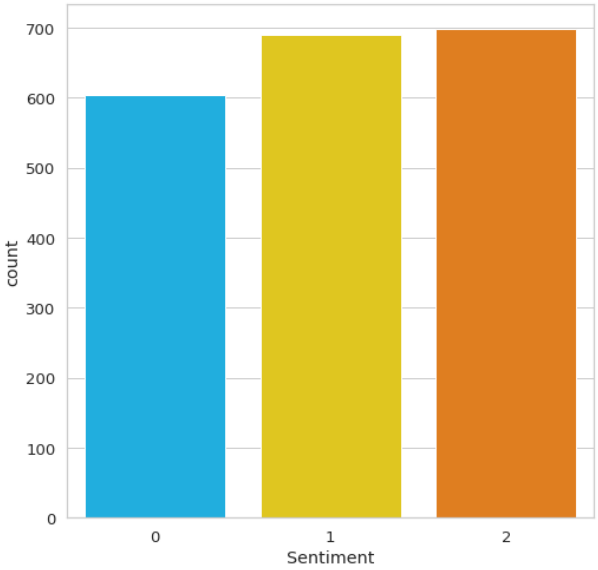

Figure 3: Sentiment vs number of images

Figure 2: Incorrect Labelling: This image was labelled as hilarious

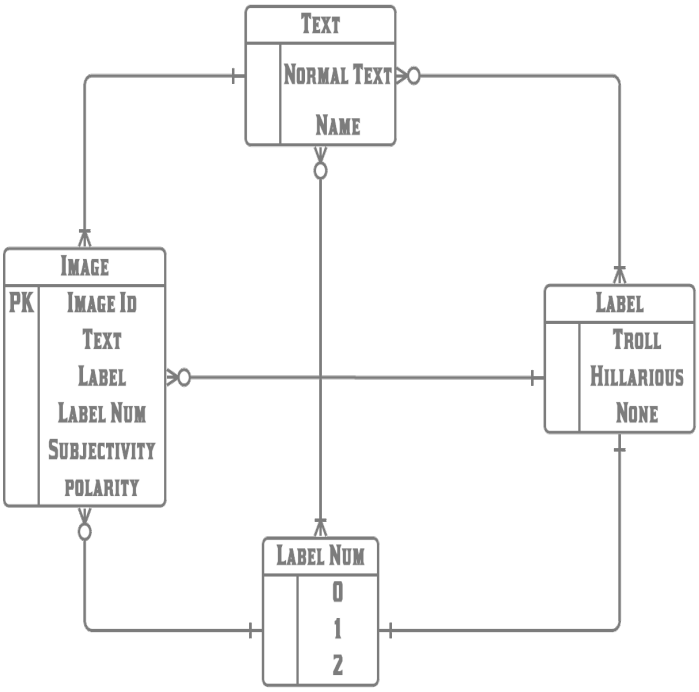

Figure 4: ER Diagram

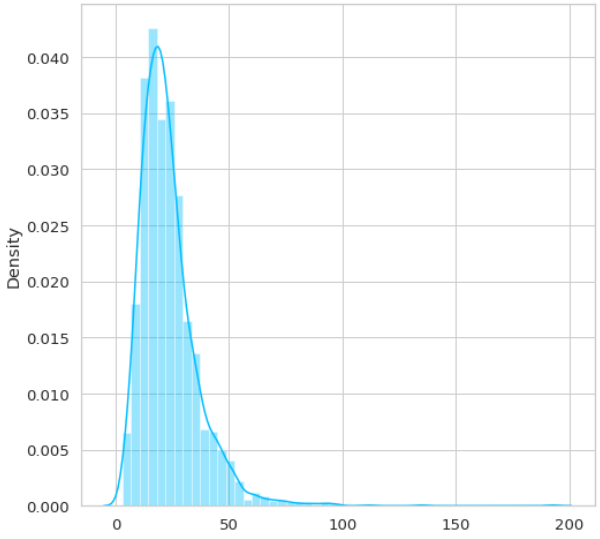

Figure 5: Distribution of the length of the tokens

ageDataGenerator class of Keras. We added Gaussian noise, rotation, shearing, zooming, etc at the time of training to make the model immune to small changes in orientation of images.

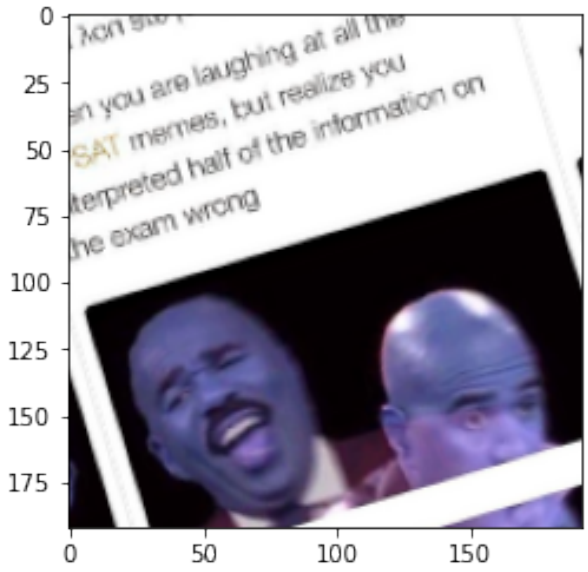

Figure 6: 15 degree rotated image

4. Models

4.1. TextBlob

We used TextBlob to get the polarity and subjectivity of each text. Using polarity and subjectivity as feature vectors, we trained a RandomForestClassifier model. The

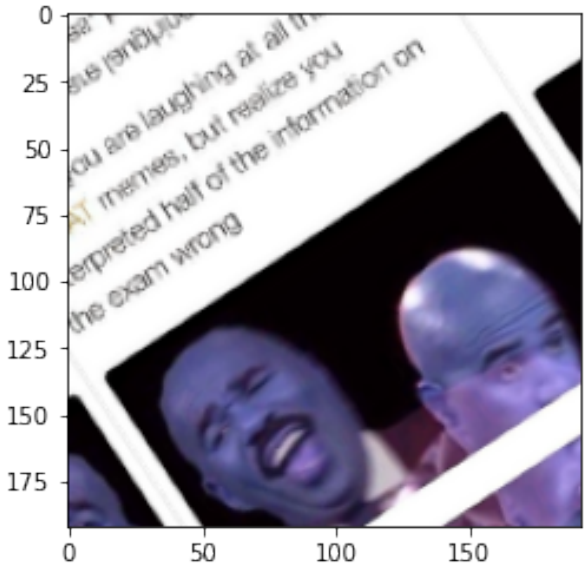

Figure 7: 45 degree rotated image

model had test accuracy of 78% and validation accuracy of 33%

4.2. FastText

We converted the text in FastText format and using train_supervised method of fasttext, we trained a model. The model predicted hilarious most of the time(90% of the time) and troll the rest (10% of the time). Hence the model was discarded.

4.3. Sentence Transformers

We used pretrained sentence transformer model **"distilbert-base-nli-stsb-mean-tokens"** to get the embeddings of the sentence using model.encode method. Taking these embedding as feature vectors, we trained a RandomForestClassifier. The model had training accuracy of 84% and validation accuracy of 38%. Taking these observations into consideration, we considered "distilbert-base-nli-stsb-mean-tokens" as our final text embedding model.

4.4. VGG 16

We used the encoder region(the first 14 layers of VGG) as the encoding generator of image. Using this encoding as feature vector we trained a RandomForestClassifier. The classifier had training accuracy of 80% and validation accuracy of 28%. We included only 10 layers of the model to encode and the validation accuracy increased to 33%. We finally took first 10 layers of VGG-16 as encoder of our image data. We also performed classification by including features of celebrities being present or

not. This was done on the observation that the celebrities like Rowan Atkinson, Neil Patrick Harris etc were more likely to appear in hilarious memes and the celebrities like Rahul Gandhi, Joe Biden, Donald Trump, etc were more likely to appear in troll memes. The model accuracy did not change significantly by adding those features.

4.5. Multimodal model for meme classification

We created a keras model that took two inputs(image input and text input). We used keras texts_to_sequences method to learn embedding of the text. We passed the embedding to an LST followed by a dense layer. A dropout layer was added to avoid over fitting. Image data was encoded using the first 10 layers of the VGG 16. The two encodings were then concatenated and passed through a dense classification layer to give predictions. We also added L1-regularization in dense layers to prevent over-fitting in the model. We added callbacks like early stopping, ReduceLROnPlateu and checkpoint to monitor whether the model was learning or not.

4.6. BERT and Transformers by Hugging Face

We trained a BERT(Bidirectional Encoder Representations from Transformers) to learn the embeddings on the text and using these embedding trained a model. The model was trained on GPU and was coded in pytorch. The model overfit(89% training accuracy and 28% validation accuracy) and did not provide significant difference in accuracy.

5. Loss function

We used Sparse categorical cross entropy as loss function in the training of the multimodal model. This loss function was specifically chosen since it did not require to one-hot encode the labels and hence was memory efficient. Also the loss function performed exactly like categorical cross-entropy loss.

6. Predictions

6.1. TextBlob

The accuracy predictions using textblob subjectivity and polarity as feature vectors was 33% on validation dataset and 78% on training dataset.

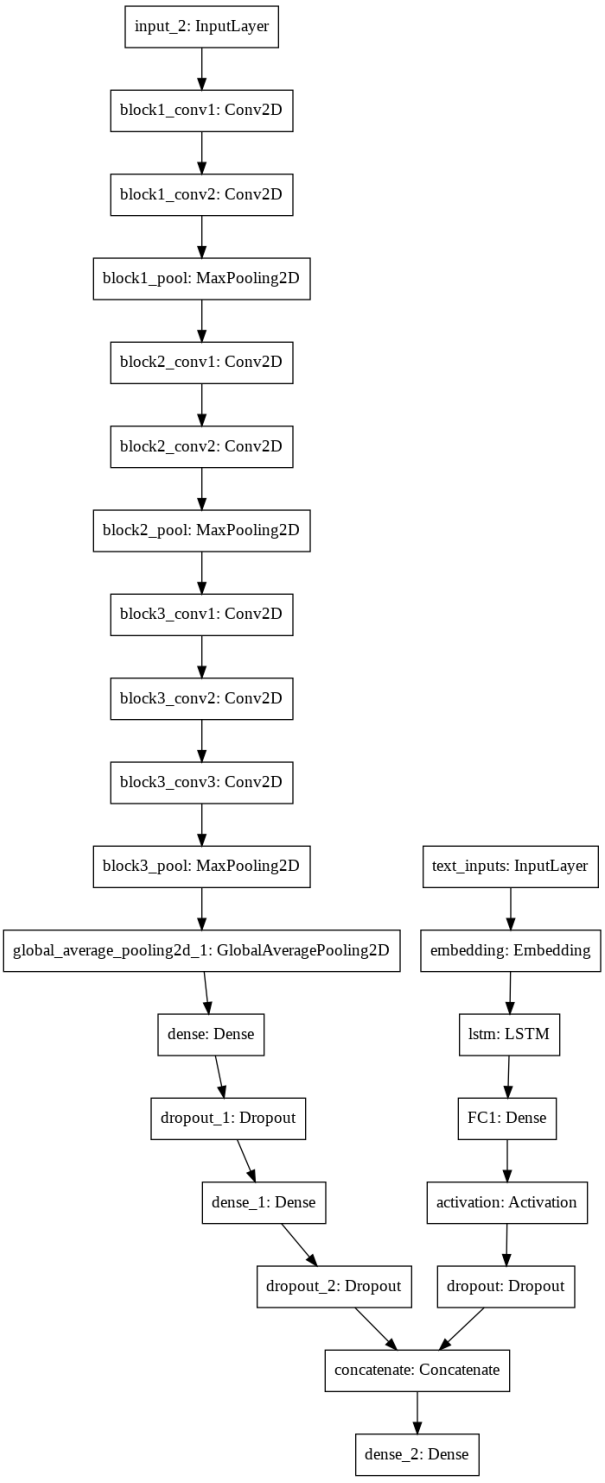

Figure 8: Multimodal Model

6.2. Fast text

The model predicted with a training accuracy of 30% and validation accuracy of 33%

6.3. Sentence TransFormer

Using the embedding given by pretrained distilbert model from hugging face, the prediction accuracy of this model on validation dataset was 38% and training accuracy was 84%

6.4. VGG 16

This model predicted with a training accuracy of 80% and validation accuracy of 33%

6.5. Multimodal model

This was the best model to predict the distribution with a training accuracy of 93% and validation accuracy of 42%. The model clearly overfits. So, we reduced the learnable parameters step by step to a mere 3000 parameters. However no significant improvement in validation accuracy was observed. As can be seen that the

Figure 9: Training and validation loss vs Number of epochs

model training and validation loss both were decreasing but validation accuracy hit a peak high of 42%.

6.6. BERT and Transformers by Hugging Face

The validation accuracy of this model was around 28% and the training accuracy of this model was 80%.

Other models and text embedding techniques were also used but there performance being very poor are not mentioned here.

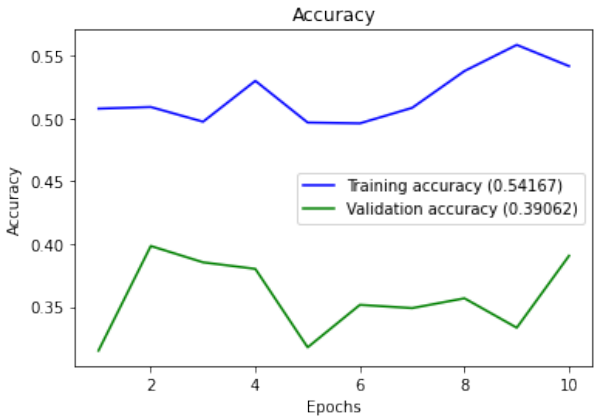

Figure 10: Training and validation Accuracy vs Number of epochs

7. Future work

The models can be improved by tuning the hyper parameters to achieve the best accuracy. Also a larger dataset can be procured to avoid over-fitting issues and the dataset annotation can be improved. We can use visual-Bert to model instead of simple CNN or VGG 16 for better sentiment analysis from images.

8. Links to Codes

TextBlob FastText Distilbert Multimodal BERT
The notebooks can be found here: 1, 2, 3, 4

References

[1] Applied Natural Language Processing with Python,Taweh Beysolow II, (pbk): 978-1-4842-3732-8

[2] Content based Imaged Classification, Efficient machine learning using robust feature extraction techniques, Rik Das, ISBN 9780429352928 (ebook)

[3] DistilBERT(2019), *a distilled version of BERT: smaller, faster, cheaper and lighter*, Victor Sanh and Lysandre Debut and Julien Chaumond and Thomas Wolf

[4] Sentiment Analysis with BERT and Transformers *by Hugging Face using PyTorch and Python*, Venelin Valkov,https://curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/

[5] BERT(2019): *Pre-training of Deep Bidirectional Transformers for Language Understanding*, Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova