Meme-emotion classification

Irada Bunyatova
Ecole Polytechnique

Victor Videau
Ecole Polytechnique

7th December 2021

Abstract

In this work, we study emotions in memes. A meme is an image with a caption spreading on the internet. Its purpose is to catch people's attention with humour, sarcasm or provocation. Therefore, memes provoke emotions. We are interested in classifying the sentiments that a meme transmits. Is the message of the meme overall positive, neutral, or is it dark and pessimistic ? We also study different types of humor, like whether a meme is funny, sarcastic or offensive. We base our work on pre-trained deep neural networks for image classification and text sentiment analysis. We finetune the models individually for each particular task. Then, we combine them to take advantage of both text and image model predictions.

1. Introduction

We tackle the problem of classifying memes according to the sentiment they convey. In the first part on **sentiment classification**, we focus on the overall feedback of people on the memes. We distinguish 3 categories : positive, negative and neutral. After inspecting the dataset we have observed that positive memes are generally funny or motivational (they inspire people) and not offensive. People would enjoy in majority such memes. On the contrary, negative memes are disturbing, it possibly makes people feel uncomfortable. Negative memes are mostly offensive, or neither funny nor motivational. Finally, neutral memes are in between.

In the second part on **humour classification**, we study 3 types of humor. We train a single model to predict whether a meme is funny or not, sarcastic or not, offensive or not. This is a multi-label classification.

Finally, we propose a **multi-task classification** model that does both the sentiment and humour classification at the same time.

The problem of meme classification is interesting in practice. In the era of social medias, there is a huge amount of online content published daily. It is necessary to have accurate recommendation systems to help users accessing relevant posts. With meme sentiment classification, a recommendation system could identify the preferred type of humor of the users based on their 'likes' and the time they spend on each meme publication. Then, it would be able to show adapted memes in the Facebook news stream for example.

2. Work we rely on

As the memes are composed of texts and images in the provided dataset, our work relies on image classification and text sentiment classification.

In the field of images, AlexNet triggered a boom in the research on convolutional neural networks in 2012. A lot of architectures were proposed for image classification, each one often improving the performance compared to previous ones. Nowadays, it is possible to use pretrained models like VGG, Resnet, ZFNet, Inception, DenseNet, etc. In this work, we decided to use and fine-tune the ResNet18 model [1] that was pre-trained on the ImageNet dataset. ResNet18 is a CNN model that uses shortcut connections in order to address the vanishing gradient problem. In addition, in the architecture of ResNet there is an average pooling layer which helps to reduce the number of parameters of the model. To illustrate, in VGGNet after convolution layers there are several fully connected layers - which results in 140 million parameters. In contrast, in ResNet18 there are 11 million trainable parameters, which significantly reduces the training time of the model.

In NLP, the transformers introduced in 2017 in the paper 'Attention is all you need' [2] revolutionized the field. Transformers outperformed all previous existing methods, which were mostly based on recurrent neural networks. To deal with the texts of the memes, we fine-tune a BERT classification model, a transformers model for NLP developed by Google in 2018 [3].

3. Description of our method

3.0 Description of the dataset

We present first the dataset we worked on. It is the Memotion Dataset 7k, available on Kaggle [4]. It contains about 7000 memes. The memes are given as images, with the caption written in black or white on the image. The caption on the image can be of any font size and at any position. Fortunately, the captions are also given as text data with the image, which avoids the need of detecting text in the image. There are two versions of the text (all in english). The first version is the texts obtained directly from the digital image with Optical Character Recognition (OCR). The second version is the corrected version of the first, where mistakes of OCR were fixed. When working on the models on the text data, we use the corrected version of the sentences. However, we note that a model used in practice would collect memes from the internet, where no ground truth for the text is available. So, OCR is the only option for a real usage of the model on new memes. In that case, an automatic correction of the sentences could be done. The last observation on the dataset is that there are multiple image sizes. With the input of convolutional neural networks being squared images of fixed size, this raises a challenge. To address the problem we first use zero padding to make an image squared [6]. Then, we resize the resulting images so that they match the input size of the ResNet18 : 224*224.

3.1 Task 1 (a) Sentiment Classification on images

We start by describing our work for the first task, which is meme sentiment classification. The target variable concerned is called overall_sentiment and has 5 categories. To simplify the problem, we merge the similar categories : 'very positive' with 'positive', and 'very negative' with 'negative'. We obtain 3 final categories : 'positive', 'negative' and 'neutral'. We observed that the dataset was imbalanced for these categories. Among the 6992 observations, there are 4160 'positive' memes, 2201 'negative' ones and 631 'neutral'. This disequilibrium needs to be handled, otherwise the training will be biased. The model will be tempted to always predict the most frequent class of the training, since it artificially gives good performances on the imbalanced train and validation sets.

We start with a model based solely on the images. We followed this pytorch tutorial [5] and adapted the parts of the code to initialize and train the model. We used a pre-trained ResNet18 model. To adapt it to our task, we replaced the output layer which originally had 1000 output dimensions by a linear layer with 3 output dimensions, one for each of our classes. As it is a multiclass classification problem, we use the Cross Entropy Loss. In pytorch, the implementation already includes the softmax function before applying the loss to the output of the model. Now, it remains to train the modified network for its new task and on the new data. We split our data into a train set and a validation set in a stratified manner, with 80% of the data used for the training. We assessed the models on the validation set. To help the model to generalize and reduce overfitting, we do data augmentation on the train images. We randomly flip vertically and horizontally the images, apply a crop with some probability in a random squared area, randomly perform color jittering and distortion (change of perspective). This increases the diversity of the train images. As it is done through a data loader, it does not increase the number of train images stored in memory. Consequently, this is not designed for solving the problem of imbalanced classes. One way to solve the problem is to take images from the underrepresented classes, disturb them with data augmentation and save the modified images in memory as new observations, along with the original ones. This would balance the number of observations per class. However, how to choose which image to augment and which augmentation methods to choose? Any choice would give a different importance for observations from the rare class, because some will be arbitrarily augmented more times than others. Another solution to this imbalance problem is to use class weights in the loss. The rarest a class is, the more penalized an incorrect prediction should be. Therefore, we use the Cross Entropy Loss with class weights. The class weights are given by the following formula :

$$w_c = \frac{m}{n_{tot} * n_c} \quad (1)$$

where w_c is the weight of the class c, m the size of the dataset (number of observations), n_{tot} is the total number of classes (3 in our case) and n_c is the number of observations of class c. On our training dataset, it makes a weight of 3.69 for the 'negative', a weight of 1.06 for the 'neutral' and a weight of 0.56 for the 'positive' classes. The problem of imbalanced datasets is complex. This solution will help the model to learn better from classes with few observations, as it will give more importance to those samples when computing the loss.

3.1 Task 1 (b) Sentiment Classification on text

Next, we used a model with the texts only. We inspired our code from the NLP course we followed at Ecole Polytechnique. We use a pre-trained Bert model, more precisely a DistilBertForSequenceClassification with 3 labels. Data augmentation is less straightforward to do with texts compared to images. The operations are more complex and thus require more time, so they should not be

applied on the fly in a data loader. In addition, the implementation of the Bert models makes it hard to use class weights in the loss for solving the problem of imbalanced data. For these reasons, our data augmentation for texts is different. From the rarest class ('negative' memes), we generate a lot of new text data by variating the existing ones. We add these new observations to our dataset in memory. By transforming each 'negative' meme text 13 times, we end up with 631+13*631 = 8834 observations for the rarest class. Similarly, we augment the 'neutral' class 3 times and we get 2201+2201*3 = 8804 observations for this class afterward. Finally, we do the same one time for the most common class, the 'positive' memes, and get 4160+4160 = 8320 samples. We augmented the classes to roughly the same number of samples to avoid the need for class weights in the loss.

Although not necessarily for the above reason, we augmented as well the most common class. This was done to also increase the diversity of the most common class, and not only the less common ones. Now, we explain how we did the text data augmentation. We use back translation, which works as follows. Consider the sentence *"You see, in this world there's two kinds of people, my friend: those with loaded guns, and those who dig. You dig."* and choose a target language, French for example. Translate the sentence from English to French using your favorite NLP model for translation. We used EasyNMT [ref]. This gives *"Vous voyez, dans ce monde il y a deux sortes de gens, mon ami : ceux qui ont des armes chargées, et ceux qui creusent. Vous creusez."*. Now, translate this sentence back to English. We obtain : *"You see, in this world there **are** two kinds of people, my friend: those who have loaded **weapons**, and those who dig. You**'re digging**."*. The words in bold have replaced some from the original sentence. Overall, they are synonyms and the meaning of the sentence is the same. Back translation introduces variability in the sentences, and it can be used with several target languages. To augment our text data with back translation, we used for example German, Russian, Hindi, Italian… We just had to make sure the target language was not completely destroying the meaning of the original sentence. For example, we observed that Japanese changes the sentence so much that only some keywords survive to the back translation. So, we tested and filtered the target languages to avoid that.

So, we enriched the original dataset to roughly balance the classes. Then, we must do the train-validation split carefully. All the versions of a meme's text after augmentation should go together in the same set, either training or validation, to avoid communication between those two sets. We do so by using an identifier for each original meme and doing the split on the identifiers. This may result in classes with a slightly different number of observations in the train set. To ensure zero bias towards class frequency in the train set, we randomly drop some observations in the classes in excess, until all 3 classes in the train set have the same number of samples.

3.1 Task 1(c) Sentiment Classification on texts and images

With the two previous tasks before we have already trained 2 models for sentiment classification - one based on images and another on texts. To make use of both data types we can combine the 2 models. For this task, we created a voting classifier. This voting classifier will be used to average the predictions of the 2 models to make the final prediction. Given a sample to classify, we provide it as input to our fine-tuned ResNet18 and BERT. As the last layer of both models is a linear layer, we will apply softmax on the output of the models. This gives us the probabilities for the two models for each of the 3 classes. Then, we average the obtained probability values for the classes 'negative', 'positive' and 'neutral'. Finally, we predict the class with the highest probability.

3.2 Task 2 (a) Humour Classification on images

The goal of task 2 (a) is to identify whether or not the provided meme is sarcastic, offensive and humorous. Originally, in the provided dataset, each of these three classes could take 4 possible values. For instance, for humour, it could be 'not funny', 'funny', 'very funny' and 'hilarious'. The values in two other classes are also representing the degree of how offensive or sarcastic the meme is. However, as the dataset is not big enough for handling such a complex problem of predicting the degree of each of the three classes, we decided to modify the values. They were modified in the following way for humour class: 'not funny' was classified as 0 and 'funny','very funny' and 'hilarious' were classified as 1. The modifications for offensive and sarcasm were performed similarly. So for each of the three classes the prediction is binary. This whole problem of predicting now became a multilabel binary classification problem. Consequently, the binary cross entropy with logits was used. The implementation of this loss function in pytorch already includes a sigmoid layer, so no function was applied on the output of the ResNet18 model. As for data augmentation in this task, it was done in the same manner as it was described in Task 1 (a).

We previously described in Task 1 (a) the importance of having balanced data to be able to train the model well and obtain non-constant predictions. However, in this task we also face a problem of imbalanced data. Here, the number of positively classified values is a lot more than negative ones (the ratio is from around 1:3 to 1:5 for each of the three classes). For this reason, we are again going to use weights for each class, in order to make the model be

penalized more for wrong predictions of minority classes. The weight values in this task were provided to the Binary Cross Entropy Loss with Logits. We provided the weights in the format expected by this loss function, which is simply computed as :

$$weight\_value = \frac{number\ of\ negative\ samples}{number\ of\ positive\ samples} \quad (2)$$

The weight value was computed for each class and an array of the 3 values was provided to the loss function.

3.2 Task 2 (b) Humour Classification on texts

The goal of task 2 (b) is to train a model on text only, that is able to give predictions for the 3 classes. As for data augmentation, we used a dataset that was described in the data augmentation part of task 1 (b). However, the data augmentation done for that task had an aim of balancing the values in the class of overall_sentiment, whereas in the current task the dataset consists of three other classes. We checked the ratios of imbalance and the values of ratio were from 1:3 to 1:5 in each of the classes. This indicated that despite the text data augmentation explained in the previous task, the classes offensive, sarcasm and humour are imbalanced. In order to balance the augmented dataset we did the following:

1. Find the most imbalanced class in that dataset, which is 'sarcasm'
2. Create a new dataset that consisted only of the data representing the minority value for sarcasm
3. Randomly sample data (same number as size of newly created dataset) from the majority class and added them to dataset from step 2

As a result, the data was balanced for the class sarcasm. We have observed that in the resulting dataset we have also reached balance (even though not the perfect balancing) for the classes offensive and humour. Finally, this balanced dataset will be provided to the BERT model. The loss used in the BERT model is binary cross entropy, which is exactly what we need for our multilabel binary classification problem.

3.2 Task 2 (c) Humour Classification on images and texts

In this task we create a voting classifier for the trained BERT and ResNet18, in the same way as described in Task 1 (c). However, in this task, instead of applying softmax function to the output of both models, we apply sigmoid on each output component, as we are dealing with binary multilabel classification. Then, we average the probability values that we obtain from models for classes humour, offensive and sarcasm. Finally, for making a prediction for each class, we compare the obtained value with 0.5, and classify it as positive if it is more or equal to 0.5 and negative otherwise.

3.3 Task 3 (a) Multi-task classification on images

In this task the goal is to create a model that is able to do both sentiment and humour classification on images. The target variables in this case are the combination of those from Task 1 and 2. However, if we just combine those labels, we will have three binary classes (sarcasm, humour and offensive) and one non-binary class (sentiment). In order to simplify the problem and computation of loss, we one-hot encoded the overall sentiment class. As a result, our dataset now consists of 6 binary classes - sarcasm, humour, offensive, negative, neutral, positive. Similarly to the handling of imbalanced data problems in previous tasks, here we again provide weights to loss function. The weight value for each of the 6 classes will be computed using the formula presented in Task 2 (a). As for the loss function, since all the classes have binary output, we will use Binary Cross Entropy Loss with Logits.

3.3 Task 3 (b) Multi-task classification on texts

For doing multi-task classification on texts we used the same method for balancing the augmented dataset as described in Task 2 (b). The resulting dataset became perfectly balanced for sarcasm, and overall roughly balanced for negative, positive, neutral, offensive and humour. This dataset was provided to the BERT model for making the predictions.

3.3 Task 3 (c) Multi-task classification on images and texts

For this final task the same implementation as described in Task 2 (c) was used. The only difference is that in task 3 after applying the sigmoid layer on output of both models and averaging the values, the predictions will be made slightly differently. For the first three values that represent binary classification of humour, sarcasm and offensive the predictions will again be made by comparing the value to 0.5 and classifying accordingly. For the last three classes that represent 'negative', 'neutral' and 'positive' overall sentiment, the class representing the maximum value will be the one predicted.

4. Results

In this section we will provide the values of accuracy for the three tasks - Sentiment, Humour and Multi-task Classification that we have trained on images, texts and images & texts. For images the best value of accuracy on the validation set obtained during training is presented. For texts we take the accuracy on the validation set when the model is trained. Finally, for classification with both images and texts, we used the voting classifier on the model trained on images (with the weights that gave the accuracy reported in the tables) and the final model trained

on texts. We note that for every model, we observe no consistent decrease of the validation loss starting from the first epoch. All our attempts of freezing layers, adding weight regularization, dropout or data augmentation have not solved that problem.

4.1 Sentiment classification

For the task of sentiment classification with three possible prediction values ('positive', 'negative', 'neutral') we obtain the accuracy values that are better than random predictions for all the types of data that we use. Due to data imbalance, the validation dataset was composed of 60% of 'positive' labels. Constant prediction could give an accuracy of 0.6 but we see the model does not fall into that trap, thanks to the weights in the loss that solves the problem of imbalanced data. We could see in the table below that the accuracy values for images and texts are roughly the same. Finally, the accuracy was increased by 2% by combining models trained on texts and images.

Table 1: Accuracy for sentiment classification

	Images	Texts	Images +texts
Accuracy	0.4896	0.4807	0.5007

4.2 Humour classification

For humour classification, the goal was to predict three binary output values for sarcasm, offensive and humour. In the table below, the accuracy is the mean of the accuracies for these three classes. We observe that for images the accuracy is 0.5286, which means that the model was almost not able to learn any characteristics from the images for predicting if it is sarcasm, offensive and humour, or not. For BERT, the performances on the validation set are better. We could expect that because text is more directly related to sarcasm, offense and humour than images are. Finally, when we combined the models for texts and images, we obtained a better accuracy than the 2 individual models, which tells us that the image classifier can help as well.

Table 2: Accuracy for humour classification

	Images	Texts	Images +texts
Accuracy	0.5286	0.5968	0.6059

4.3 Multi-task classification

In this task we have more values to predict - as this is a combination of the two previous tasks. In table 3, the values of accuracy are provided as an average of the accuracy values for the overall sentiment, humour, sarcasm and offensive classes. For images, we have obtained a mean accuracy of 0.5536 and for texts, a value of 5729. Overall the text gives more information to predict the humour types, which counts as 0.75 of the mean accuracy. Though, we observe again that combining the two models increases the accuracy by around 2% as in Task 1.

Table 3: Mean accuracy for the multi-task classification

	Images	Texts	Images +texts
Accuracy	0.5536	0.5729	0.5905

4.4 Comparison of Tasks 1-2 with Task 3

In order to be able to compare the accuracy values in Task 3 to the values in Tasks 1 and 2, we have computed the validation accuracy of the model trained on images and texts for each of the 4 classes, as shown in Table 4:

Table 4: Detailed accuracy for the multi-task classification with images and texts

	Sarcasm	Humour	Offensive	Overall sentiment
Accuracy	0.6338	0.7225	0.5322	0.4735

The average value for the sarcasm, humour and offensive accuracy given in Table 4 is 0.6295. It is better than the mean accuracy of 0.6059 obtained in Task 2. This indicates that the extra target variable 'sentiment' can help the model to predict the types of humour. For the overall sentiment, the value obtained in Task 3 (0.4735) is less than for Task 1 (0.5007). This can be explained by the high variability of the model training with such a small dataset.

5. Conclusion

In this project we have been working on tackling quite a hard task of being able to predict the various classes by training models on a small dataset. We have implemented various techniques to overcome the problem of imbalanced data, as well as to augment the dataset. The results we obtained show that a few thousands observations are not enough to fine-tune deep learning models for tasks involving complex data such as texts or high resolution images.

[Collaboration policy]

Overall we did all parts together

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[4] Sharma, Chhavi, Paka, Scott, William, Bhageria, Deepesh, Das, Amitava, Poria, Soujanya, Chakraborty, Tanmoy, Gamback, and Bjorn. 2020. Task Report: Memotion Analysis 1 @SemEval 2020: The Visuo-Lingual Metaphor! kaggle.com/williamscott701/memotion-dataset-7k

[5] Nathan Inkawhich. Finetuning Torchvision Models. pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html

[6] code at https://discuss.pytorch.org/t/how-to-resize-and-pad-in-a-torchvision-transforms-compose/71850/2