Machine Learning Algorithm Predicts The Popularity Of A Meme

Robin Leman

MAIS 202 - Winter 2021 - Deliverable 2

Abstract

In this experiment, a machine learning algorithm will try to predict "how funny" a meme is. By training the algorithm on a dataset of memes from Reddit, the machine will be able to learn the popularity of a meme based on its number of upvotes. Then, it will be able to predict how funny an original meme is.

1 Data Preprocessing

The selected dataset is the Reddit Memes Dataset found at `https://www.kaggle.com/sayangoswami/reddit-memes-dataset`.

From the dataset, we had to extract both the meme image and its associated popularity score.

Each image was either pre-downloaded or accessible through a Web API. We used both approaches to minimize data loss. Then, they were pre-processed by being first converted to an RGB format, since some of them were png files and other jpg files, and then resized to a 512x512 format. Finally, we converted each images to an array and normalized each pixel value in the range $[0; 1]$.

The popularity score is defined by #upvotes - #downvotes and was directly extracted from the json. For the first deliverable, we computed the popularity score mean: each meme below average were declared "boring" and each meme above average were declared "funny". The two classes were respectively represented by the integers 0 and 1.

2 Model

A Convolutional Neural Network was used to train the algorithm on the set of images and labels. The implementation was made through the tensor flow library on python. This library was used due to its large amount of documentation and tutorials.

We used the standard applicable model of image classification to train the algorithm[1]. After dividing the dataset between training and validation with a 0.8 ratio, each set was cached, shuffled, batched and prefetched for optimization.

We used the Adam optimizer and a Sparse Categorical Crossentropy loss function. Both are standard for image classification. The training was run on 3 epochs.

3 Preliminary Results

The training was made on 80% of the dataset and the remaining 20% were used for validation. The result of the three epochs is described below;

1. loss: 3.99 - accuracy: 0.59 - validation loss: 0.51 - validation accuracy : 0.92
2. loss: 0.65 - accuracy: 0.60 - validation loss: 0.42 - validation accuracy : 0.92
3. loss: 0.59 - accuracy: 0.69 - validation loss: 0.55 - validation accuracy : 0.73

4 Discussion

As a first step, the model was trained with only two classes. This means that if the algorithm were to guess at random, it would have an accuracy of 0.50. The result has to be interpreted relative to this value.

Our training accuracy is increasing and our training loss is decreasing. This means that the model is, as expected, becoming more and more able to predict what is funny or not. However, the difference between the accuracy of a random guess and the one of model is too small to draw conclusions.

The validation accuracy for the first two epochs is surprisingly high, with a 0.92 value. This value is far from the training accuracy and is unexpected. Indeed, too many characteristics make a meme funny and it is not expected from a model to understand all of them with only 3227 images. To understand this value, the testing dataset was analyzed; it only contained boring images. From that, it was discovered that our dataset was originally sorted, and even if each dataset were individually shuffled, the splitting between testing and validation was ordered.

The third epoch inverses the trend of our validation result. The validation loss increases and the validation accuracy decreases. This suggests an overfitting of our training dataset diminishing the accuracy of our validation step.

5 Next Steps

There is still a big margin to improve the model. First, we can add more labels to our dataset. The idea would be two divide the dataset according to the popularity score quartiles. The first quartile would be labeled "boring", the second one "meh", the third one "funny" and the last one "MAIS worthy". More classes would increase the gap between the accuracy of a intelligent guess and the accuracy of a random one. Our accuracy will thus be more significant.

The dataset also needs to be well shuffled before being split into a training and validation set. As my favorite cookies recipe said; "stir well".

The model can also be improved by giving it, in addition to the image, the meme caption as plain text. The meme caption is central in making the meme funny and should have an important weight in the training.

6 Conclusion

To conclude, our model is for now unsatisfactory, and finer tuning needs to be done in order to draw conclusions from it.

However, the goal of this experiment is not to reach perfect accuracy. The characteristics that make a meme funny are too broad and too numerous. It will depend on its social and historical context, timing, subreddit and audience. The goal of this experiment is more about the concept of teaching a machine humour and see if and how a machine can laugh.

I'll let you on this quote from The Imitation Game[2];

"Of course machines can't think as people do. A machine is different from a person. Hence, they think differently. The interesting question is, just because something thinks differently from you, does that mean it's not thinking? Well, we allow for humans to have such divergences from one another. You like strawberries, I hate ice-skating, you cry at sad films, I am allergic to pollen. What is the point of different tastes, different preferences, if not, to say that our brains work differently, that we think differently? And if we can say that about one another, then why can't we say the same thing for brains built of copper and wire, steel?"

References

[1] *Tensorflow Tutorials - Load Images*. URL: https://www.tensorflow.org/tutorials/load_data/images.

[2] Morten Tyldum. *The Imitation Game*. 2014.