Meme Caption Generation Using Deep Learning

May 10th, 2020

1 Introduction

Memes have begun to sculpt political opinion to a great extent in today's age, thus making our problem of meme text generation very interesting. In our project, the deep learning model generates caption for a meme [5, 2, 4, 7]. Secondly, hilarity and relevance classifiers in this project assess if the generated caption is funny[1, 3], and whether or not the generated caption is relevant to the meme. All code can be found at the project repository here

2 Novel Aspects

1. In the input, image labels are also included along with the images, which provides context.
2. Hilarity and relevance classifier models objectively evaluate the meme generation performance.

3 Technical Approach

3.1 Problem Definition

This project tackles a text generation and evaluation composite problem. The generation part involves taking in an image, its label, and generate a sentence. The evaluation bit is a classification problem, where the categories are 'funny' and 'not funny' for the hilarity measure, and 'relevant', and 'irrelevant' for the relevance classification aspect. The criterion for evaluation were perplexity and cross-entropy loss.

3.2 Experimental Settings

All codes were run on CS data.purdue GPUs. The code is in Pytorch. **Dataset 1: Data for the Meme Caption Generator**
The data can be found in the following repository.
Total size of the dataset is 414,380. Each image in the dataset has a unique label, plus approximately 160 caption sentences. There are 2505 images in total. Splitting ratio for train/validation = 70/30. Thus, in total 290k datapoints are used for training, and 124k datapoints for validation.

Dataset 2: Data for the Hilarity Classifier
The dataset consists of English one-liners. Each datapoint belongs to either of two classes: Serious vs Hilarious. 50,000 samples used for training with a 75/25 training-validation split. 25000 caption sentences used from Dataset 1 as funny text; 25000 ABC news headlines from Kaggle used as serious text. Average GloVe word embeddings [6] were used to represent each datapoint.
Dataset 3: Data for the Meme Caption Relevance Classifier
This contains a total of 16000 datapoints consisting of two classes: Caption Relevant to Meme vs Caption not Relevant to Meme. To curate the data, we obtained 40 meme templates from MemeProject github repository. For every meme template obtained 200 associated captions from Memeproject github repository. These 8000 samples formed the Relevant dataset. Also, for every meme template 200 random meme captions were scraped from imgflip. These 8000 samples formed the Irrelevant dataset. Average GloVe word embeddings **(year?)** were used to represent each datapoint.
Only caption sentences of length ≤ 32 words were included. General text preprocessing like lowercasing, punctuation and special character removal were performed. $< START >$ and $< END >$ tags were added to beginning and end of every caption sentence. For words in the dataset that didn't exist in the GloVe vocabulary, I assigned a random 300-sized embedding. For image preprocessing, normalization, resizing to 224*224 pixels, and horizontal flip was performed.

4 Details about model

4.1 Meme Caption Generator: Encoder

The image is embedded via pretrained VGG16 plus a fully connected trainable layer. The label words are embedded via GloVe and averaged. Th two resulting vectors are concatenated, and passed through a Neural Network to output a 300-sized vector. This acts as the hidden state for the Decoder.

4.2 Meme Caption Generator: Decoder

The Decoder is a unidirectional LSTM that accepts the above 300-D vector as hidden state. Each cell accepts the GloVe embedding of a word in the caption sentence as input, and outputs a softmax probability distribution vector of the size of the dataset's vocabulary.

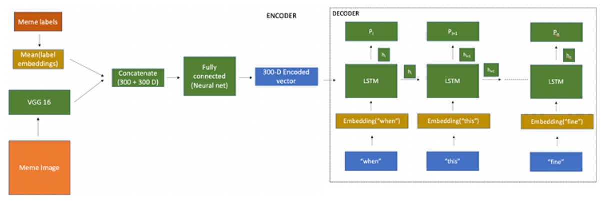


Figure 1: Meme Caption Generator Model Architecture

4.3 Hilarity Classifier

Hilarity Classifier evaluates if the caption generated by the encoder-decoder model is funny (output 1) or not (output 0). The data comprises of two classes : (i) Hilarious meme captions and (ii) serious newspaper headlines. To classify between these two classes we used a Neural network architecture with a single hidden layer and Sigmoid activation. The number of neurons in the hidden layer was optimized using cross validation and set to 1000. Similarly the learning rate was optimize to 0.01.

4.4 Meme Relevance classifier

This classifier evaluates whether a given caption is relevant to a given meme template (outputs 1) or not (outputs 0). It takes a caption and a meme template (by name) as input. The data belongs to two classes : (i) the meme template is relevant to the given meme or (ii) it is irrelevant (randomly assigned) to this meme. To classify between these two classes we used a Neural network architecture with a single hidden layer and Sigmoid activation. The number of neurons in the hidden layer was optimized using cross validation and set to 1000. Similarly the learning rate was optimized to 0.01.

5 Experimental Settings: Hyperparameter Tuning

5.1 Meme Caption Generator

Hyperparameter tuning was done to decide the optimum learning rate, dropout rate etc. The final learning and dropout rates decided were 0.005, 0.5. We've shown the learning curves for learning rates herewith. We'll use perplexity and hilarity as two metrics to evaluate our model in this project.
Here, perplexity is a measure of the inverse probabilities of predicting the next word in the sentence
Training was performed for 3 epochs. Loss and perplexity saturation was observed after the first epoch.

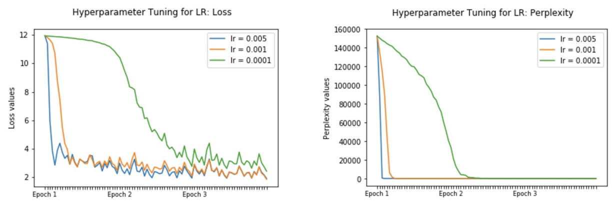


Figure 2: Loss and Perplexity learning curves of Meme Caption Generator while tuning the learning rate

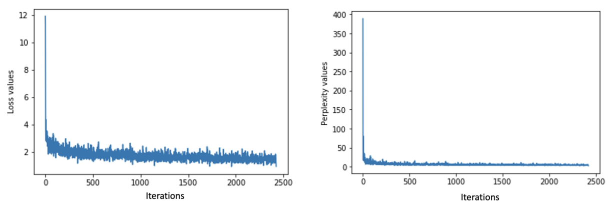


Figure 3: Loss and Perplexity of Meme Caption Generator during training

5.2 Hilarity and Relevance Classifiers

Hyperparameter Tuning: Figure 4 shows the learning curve for both models
Hilarity Model: Learning and dropout rates set to 0.01; hidden layer was set to 1000 neurons. Validation accuracy of 93% achieved. Figure **??** shows the learning curve
Relevance Model: Learning and dropout rates set to 0.01; hidden layer set to 1000 neurons. Validation accuracy of 82% achieved

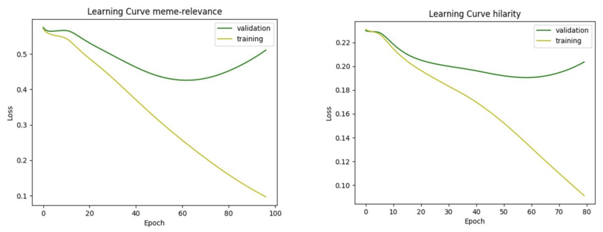
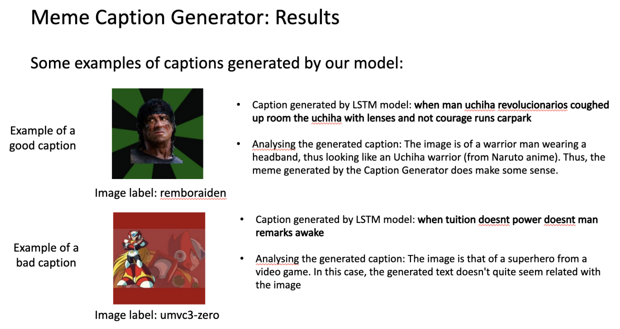


Figure 4: Learning curves for Hilarity and Relevance Classifier Models

6 Results



Meme Caption Generator: Results

Some examples of captions generated by our model:

Example of a good caption

Image label: remboraiden

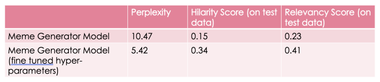
- Caption generated by LSTM model: **when man uchiha revolucionarios coughed up room the uchiha with lenses and not courage runs carpark**
- Analysing the generated caption: The image is of a warrior man wearing a headband, thus looking like an Uchiha warrior (from Naruto anime). Thus, the meme generated by the Caption Generator does make some sense.

Example of a bad caption

Image label: umvc3-zero

- Caption generated by LSTM model: **when tuition doesnt power doesnt man remarks awake**
- Analysing the generated caption: The image is that of a superhero from a video game. In this case, the generated text doesn't quite seem related with the image

Figure 5: Examples of some generated captions



	Perplexity	Hilarity Score (on test data)	Relevancy Score (on test data)
Meme Generator Model	10.47	0.15	0.23
Meme Generator Model (fine tuned hyper-parameters)	5.42	0.34	0.41

Figure 6: Evaluation of captions generated by Meme Generator model

One of the aspects of the project was to see how well the model performs when generating funny captions. As humor is highly subjective, two human testers were used to read the generated captions and give their opinion on them being funny or not. On the average, the 'accuracy' of a caption predicted to by funny and rated funny by humans was 65%. This is not a concrete measure however, as the generated captions included the non-dictionary words which the human testers mostly did not find humorous.

7 Summary

A meme generating model using encoder decoder architecture was trained. To assess the model, we trained hilarity model that predicts how hilarious a meme is and, a relevance model which tells how relevant a meme is given a label. Generator model got perplexity score of 5.42, relevance score of 0.41 and hilarity score of 0.34 on test dataset. The model can be trained for more number of epochs (it takes 18 hours per epoch, so training was time constrained). Perhaps, introducing the concept of attention on images and label will help. Training one's own CNN model for image embedding, or the ResNet model might also help improve results.

References

[1] Jim Cai and Nicolas Ehrhardt. Is this a joke? 2013.

[2] Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. Stylenet: Generating attractive visual captions with styles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3137–3146, 2017.

[3] Rada Mihalcea and Carlo Strapparava. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[4] Jonghwan Mun, Minsu Cho, and Bohyung Han. Text-guided attention model for image captioning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[5] V Peirson, L Abel, and E Meltem Tolunay. Dank learning: Generating memes using deep neural networks. *arXiv preprint arXiv:1806.04510*, 2018.

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[7] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*, 2015.