Meme Generator Using Deep Neural Networks: A LIGN 167 Course Project

Xinting Wang

University of California, San Diego | xiw018@ucsd.edu

Abstract. This project is inspired by the Dank Learning Project from Stanford CS224n 2017 by Abel Peirson and Meltem Tonulay. I replicated the meme generation system using a pre-trained inception-v3 network to process images, which are then passed into an attention-based LSTM to create corresponding captions inspired by the Show and Tell Model. The evaluation is based on perplexity scores and human assessment (the original author created a meme generator app and recorded user reaction which you can find here: https://danklearning.com/). The goal is to produce original but hilarious memes that are indistinguishable from human-made ones.

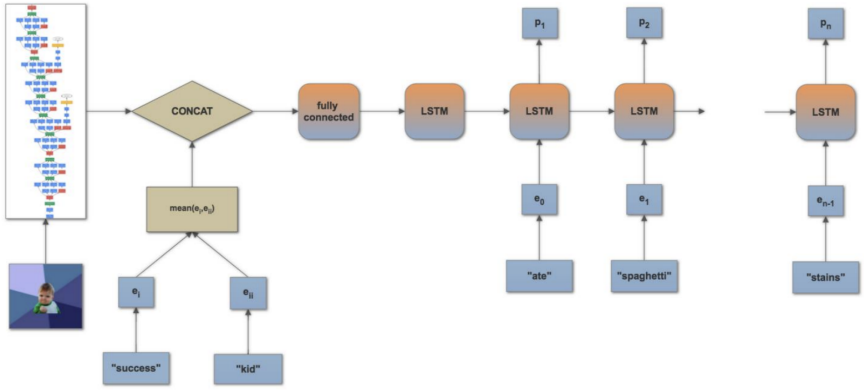
1 Introduction

Memes are the heart and soul of today's social media platform: they come in light-weighted yet packed with life philosophy and a laughable attitude. They can be penetratingly harsh and unbearable in revealing the truth, or they can gently touch your heart with nothing but a kitten meowing like a king. Meme is not unique in the family of image-and-caption, but it has captured our generation like nothing before. I was tempted by the intention to study memes on a language level and maybe unlock the secret of linguistic humour along the way.

I approach the problem in the same way that the original paper approached it, which only take into consideration of memes with captions. The predictive task here is given an image (no caption, no label), try to produce a humorous line that is relevant to the picture. I applied an encoder-decoder image captioning system similar to Figure 1: consisting of CNN image embedding initial stage, followed by LSTM RNN for language generation, then evaluated by perplexity score which has high correlation with BLEU score [1].

In my replication, I used the same data set (scraped from https://memegenerator.net/), the same image processing tool (inception-v3 model), and the same evaluataion schema. I rebuild my data processing and training in one JupyterNotebook under im2txt folder. The original project was built in TensorFlow, so I used TensorFlow for my modification as well. Please be aware that the data set contains 4 million entries so running the code could potentially take a while.

Fig. 1. Meme Generator Project Overview



2 Background

2.1 Image2Seq Models

In order to generate captions, one has to "understand" the meaning of the underlying picture, where image captioning models come into play. Traditional image classification(ImageNet) as well as advanced Encoder-Decoder scheme(Image2Seq inspired by seq2seq translation process) were considered. Deep Convolutional Neural Network(CNN) such as Alex Net and Inception Net were compared in terms of end-point evaluation accuracy. The author (and I) came to the conclusion that Inception Net worked better because its a better image classification network, but the alexnet method is slightly faster and simpler. It is also notable that the result could be further improved by using bi-directional LSTM, while whether it's humorous still open to discussion.

2.2 Recurrent Neural Network and Attention

RNNs are known to be the state-of-the-art technique used in sequential NLP projects. Here I chose LSTM with "gating mechanism" which enjoys the reputation of providing satisfying results. Attention mechanism is also introduced in this project. The author chose to proceed with Luong's attention mechanism [2](which is a modification to Bahdanau's [3]). I kept their modification with attention mechanism as one of the model variant. [1]. Here is the mathematical representation where f is the forget gate, i the input gate, o the output gate, m the memory output and W the trainable matrix:

$$
\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1}) \\
f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \\
o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{cx}x_t + W_{cm}m_{t-1}) \\
m_t &= o_t \circ c_t \\
p_{t+1} &= \text{softmax}(m_t)
\end{aligned}
$$

3 Data Set

I used the same data set used by the author, which was scraped from https://memegenerator.net/ using scraper.py. This data set contains 181 pages * 10 most-popular image templates, each has 15 most-hit captions per page * 15 pages = 407250 entries. Each entry has a image, a label (template name, such as Success-Kid, One-Does-Not-Simply etc), and a caption(meme name, such as one-does-not-simply-one-does-not-simply-read-a-meme-in-your-own-voice etc). All 4 million entries were stored in /im2txt/meme and /im2txt/Captions.txt.

4 Model

4.1 Encoder

Given predictive task here, when a meme template is run through the Inception Model, I kept the fixed length vector (image embedding) in train.tfrecords. The author proposed three possible variants:

1. Image embedding of the templates: feed template images into inception network to get the variant.
2. Word embedding of the labels: take the average word embedding of the label using pre-trained GloVe Vectors. Concatenate word embedding to image embedding directly to get the variant. In absence of a label, the author proposed a Inference-and-Beam-Search method, which starts caption generation with random k in 100 most probable words in the language model. Instead of tuning the probability model to be greedy (which search for the most probable k words), I tuned the temperature to 10 which corresponds with a more "random" selection to increase humor [1].
3. Modification based on Luong attention mechanism: instead of direct concatenation, another fully connected LSTM was placed before decoder network, taking image embedding as the initial state and feed in label GloVe embedding to get the variant.

4.2 Decoder

The decoder is a unidirectional LSTM based on the show-and-tell model. The author introduced pre-trained GloVe embedding instead of randomly initialized word vectors. Unlike the author who finalized his work using a 3 layer LSTM, I used single layer LSTM for the simplicity of computation.

5 Training

Instead of different layers of LSTM, I used single-layer LSTM decoder network and optimized with SGD Optimizer. This is all set in configuration.py in /im2txt. I tuned initial learning rate and Inception v3 parameters in configuration.py as well. All training and evaluation was done in LIGN167-code.ipynb in /im2txt. For simplicity, I only displayed one model. Fig 2,3,4 are some relatively good predictions: (at the end)

6 Evaluation

Define Perplexity Score (PP) as below: this is implemented in evaluation.py in /im2txt. Evaluation results are kept in eval.tfrecords. Other than Perplexity Score, human identification with "sense" and "humor" is also wanted. In that case, human evaluation requires more resources thus not implemented here.

$$PP(C) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1...w_{i-1})}}.$$

7 Conclusion

In this paper, I tried my best to re-implement the Meme Generator created by Peirson and Tolunay. If given more time (, teammates and computing power), I think the model could be optimized by using more layers (as the authors did) and more variants. One addition is their app (Dank Learning) which collects human reaction to generated memes. This can be an after-step to this NLP model, introducing a new feature named "Acceptance Rate". One major defect here is that humor is still not assessed (perplexity score does not evaluate humor) and the breakpoints of the slide show above were chosen manually. There are other issues such as sexism and racism to be addressed further as well. As a personal project, I'd like to re-factor the TensorFlow code into PyTorch later.

References

1. Peirson, Tolunay: Dank Learning: Generating Memes Using Deep Neural Networks http://web.stanford.edu/class/cs224n/reports/6909159.pdf
2. Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015).
3. Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).
4. Special Thanks to all kinds of TensorFlow Tutorials, especially http://warmspringwinds.github.io/tensorflow/tf-slim/2016/12/21/tfrecords-guide/

Fig. 2. /Users/summ7t/MemeProject/im2txt/memes/raptor.jpg
why are they called "gentlemen's clubs" if there's mostly women in them

Fig. 3. /Users/summ7t/MemeProject/im2txt/memes/grumpy-face-cat.jpg
i used a pc once it was awful

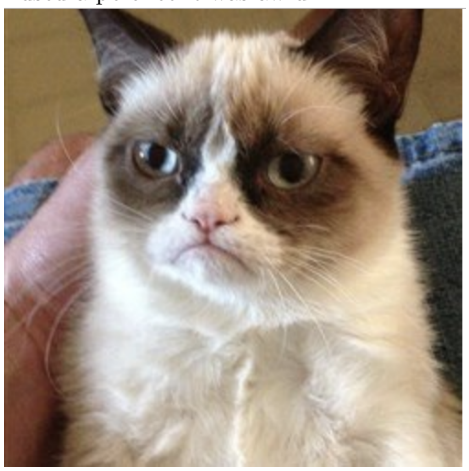


Fig. 4. /Users/summ7t/MemeProject/im2txt/memes/celestia.jpg
im a princess are you a princess too?