Meme Caption Categorization

Charles Fauman

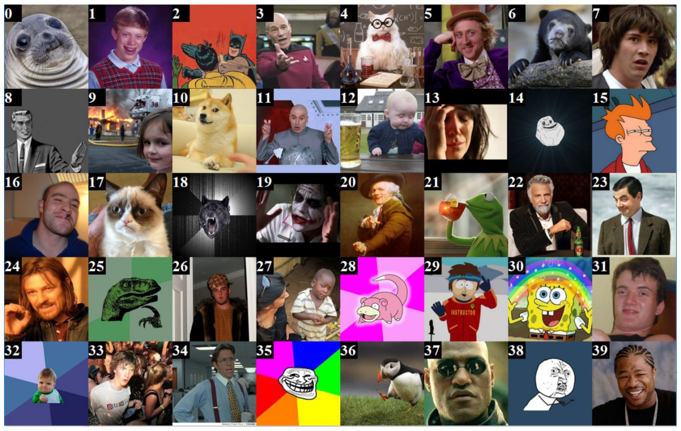
Abstract

A lot of popular social media platforms, such as imgur, reddit, and snapchat, revolve around images with text captions. This is because internet memes can be greatly enhanced by a related image. As online culture rapidly shifts, common images in memes pop up and go down , with more popular ones staying alive longer. In this paper, I explore how moderately popular memes can serve as labeled data. As memes are such a widely used form of communication, there is a massive amount of open source information waiting to be analyzed. The goal of this work was to apply text classification to this new domain using a simple deep neural network model. Then, using this model, I was able to build a simple python gui that takes in any input sentence and returns the best matching memes for it. The idea behind it is that people not caught up in currently popular, trending memes can still enhance their text through a related image.

1. Data Preparation

1.1. Collection

I took 1035 captions from each of 40 of the top meme categories from https://memegenerator.net/memes/popular/alltime
The caption for each meme is embedded in the html, meaning that I do not have to do ocr on the images.



1.2. Cleaning

I first use a language check check tool that corrects small errors such as one letter typos of common words, or a missed space between common words. I then lowercase all the text. After this, I use nltk to get word tokens, part of speech tags, and lemmas. Using the tokens and part of speech tags, I got token based positive/neutral/negative sentiment from a table produced by (Wilson et al., 2005).

2. Methods

2.1. Preprocessing

I obtained 41,400 memes, and split these at 80% per meme to 33,120 training, 8,280 testing. Based off the training set, I used word2vec to find embeddings of length 50 with a min count of 2. I then vectorized my training and testing data using these embeddings and one hot encodings of the part of speech tags and sentiment. To not have to deal with variable length memes, I made a cutoff/zerofill length of 50 tokens. Any out of vocabulary word is zeroed, but counts towards this 50 token mark as its part of speech and sentiment and placehold are kept

2.2. Model

I used keras to create my model. It consists of a .1 dropout layer into an LSTM layer with .5 recurrent dropout, with 40 output fed into fully connected dense layer with softmax activation with 40 output.

2.3. Training

I compiled the model with categorical cross entropy loss using the Adam optimizer. I then trained it for 300 epochs with a batch size of 200.

3. Results

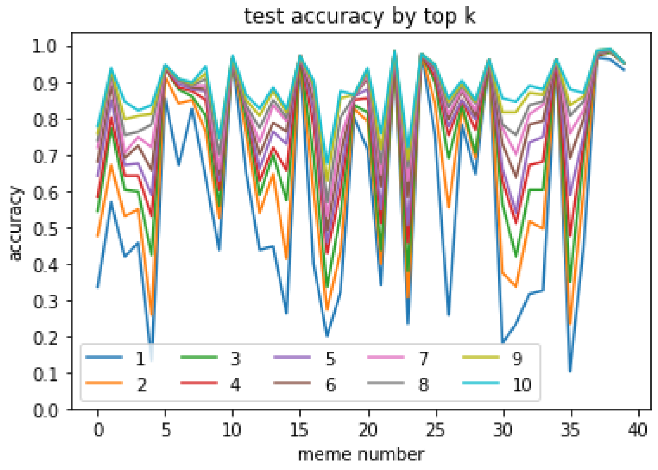
3.1. Accuracy

k	mean	std. dev	max	min
top 1	0.570	0.284	0.976	0.102

top 2	0.662	0.237	0.981	0.233
top 3	0.717	0.200	0.981	0.335
top 4	0.758	0.169	0.985	0.427
top 5	0.785	0.151	0.985	0.456
top 6	0.813	0.129	0.985	0.490
top 7	0.834	0.114	0.985	0.529
top 8	0.851	0.100	0.985	0.568
top 9	0.869	0.086	0.990	0.626
top 10	0.882	0.076	0.990	0.675

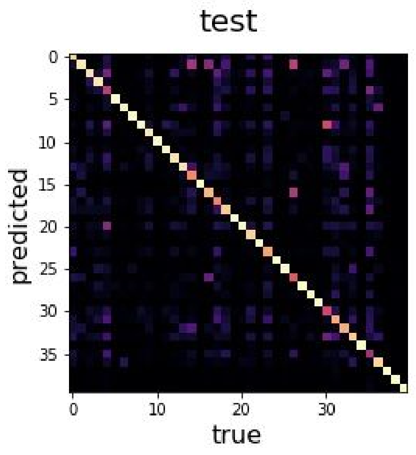
as seen in the table above, I obtained a 57% accuracy on testing data. Keep in mind that although most classification tasks in natural language processing deal with a small number of categories, in this exact task there were a total of 40 categories to choose from, meaning that a random guessing baseline would lead to a 2.5% accuracy. also did an analysis on accuracy based on a simple linear discriminant analysis model fed a flattened version of the input data. This model effectively splits along embedded word counts, and obtains a 50% accuracy on the testing set. This means that the LSTM model has a 7% gain in accuracy as compared to a naive model.

3.2. Variation in Predictive Power



To understate it, some memes performed significantly better than others. This is because of specific phrases used in memes that are unique to them. some examples are 39 ("Yo Dawg"), and 38 ("Y U No"), where that text appears at the beginning of nearly every example. Other memes, such as 35 "Trollface", seem to not have any particular patterns of words that my system is able to pick up on.

3.3. Errors



The above graph is a confusion matrix of the data, normalized along true labels. The whiter a pixel, the more it was chosen for that column, and the more purple the inverse. It can be seen that when a meme was guessed incorrectly, there were specific substitutes that the model prefered. One such example can be seen between the memes 16 and 26, pinkish-purple dot show on both (16, 26) and (26, 16). These are the memes "Good Guy Gary" and "Scumbag Steve". These memes are very similar in context, except for the fact that one has a positive connotation while the other has a negative one. Here is where a good sentiment analysis would have been able to distinguish these memes.

4. References

Theresa Wilson, Janyce Wiebe and Paul Hoffmann (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proceedings of HLT/EMNLP 2005, Vancouver, Canada.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado and Jeffrey Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. *CoRR, abs/1310.4546*, .