InspiroBot 2.0 and Generating Inspirational Posters

Kyle Wang CU Boulder

Nicole Dong CU Boulder

Mike Donovan CU Boulder

Abstract

Inspirobot (https://inspirobot.me/) is a website that uses some unknown AI strategy to generate often nonsensical inspirational posters. We consider the task of replicating this unique inspirational quote generation through three separate means: a simplistic synonym replacement baseline model, an n-gram model, and a recurrent neural network (RNN). Though it is fairly difficult to evaluate text generation, we find that the results from the RNN can at times be passable as human quotes and the probabilistic n-gram imitates the brand of Inspirobot quotes very well.

1 Why This Project?

There are two tasks to our project, generating quotes and placing them on to posters. The major job is of course the quote generation. For this, we want our generated quotes to fit some standards. First and foremost, the generated quotes should be unique in the sense that they do not appear in our dataset. Secondly, our quotes should be both grammatically correct and have an inspirational sentiment to them. The quotes do not necessarily have to make logical sense as many of the quotes generated by the Inspriobot are mostly nonsensical. Preferably, our quotes are similar to the Inspirobot as humorous pseudo-inspirational quotes. Finally, we want to randomly place our quotes on posters similar to the Inspirobot.

For quote generation, we focus mainly on two examples of language models, n-gram models and recurrent neural networks. A language model is a system that predicts what word comes next in a sequence of text. More formally, given a sequence of words $x_1, x_2, ..., x_{t-1}$, a language model computes the probability distribution for the next word, x_t:

$$P(x_t \mid x_1, x_2, ..., x_{t-1}).$$

Each of these words, x_i, must be from some vocabulary set of words $V = \{w_1, w_2, ..., w_{|V|}\}$. From our survey of online materials and papers, we see these two models seem to be the go-to strategies as they are extremely straightforward and surprisingly powerful.

2 What We Did

Here is the Github repository containing all the code we wrote for this project: https://github.com/mido3801/InspiroBotV2. Furthermore, here is a link to the survey we used to attempt to evaluate the images we generated: https://forms.gle/hEW8PGANYcXDSZan9.

2.1 Data Set & Data Cleaning

Our dataset was a dataset obtained from Kaggle of quotes of various sentiments, we specifically used the subset of quotes denoted as "inspirational". These quotes were originally obtained from a scraping of websites such as GoodReads which aggregate quotes from authors and historical figures. However, on some websites included in the scraping, users are able to upload their own quotes which leads to some questionable reliability and an increased necessity for data cleaning.

In addition to the occasional user-submitted erroneous quote there were also other things that led to the data needing to be cleaned. Within the set there were some quotes that utilized various encodings such as ISO-8859-1, while we were trying to read the file using UTF-8. In some of these instances we were able to recover the data by replacing certain ISO characters with their UTF equivalents, however in some cases the entire quote was read as garbage:

3181 Ã˜ÂºÃ™â,¬Ã™â,¬Ã™ÂÃ™â,¬Ã™â€ Ã˜Â§Ã™Æ' Ã™Å½ Ã™ÂÃ™â€° Ã™â€ Ã™ÂÃ˜Â³Ã™â,¬Ã™ÂÃ™â,¬Ã™Æ'Ã™Å½ Ã˜Å' Ã™Ë†

In general, we believe data points like the above are okay to remove as they are most likely not in English. As most character encodings are extensions to ASCII, we expect that English quotes should have readable ASCII characters. This brings us to the next issue with our data which is that there were also quotes in the dataset that were in languages other than English, such as Filipino. To remove these quotes, we compare the words in a quote to an English dictionary set provided by NLTK. If words are not in this dictionary, we remove that data point. However while removing these non-English quotes, we also potentially removed English quotes that contained slang not in the wordset, as well as quotes that had punctuation removed and words concatenated together, e.g. "end. The" became "endThe".

After all of our cleaning efforts our dataset of originally about 33,000 quotes became about 11,000 quotes. Despite this extreme drop in the size of our data, we felt that it was better to be safe with our data. In addition, we felt that 11,000 quotes was more than enough data to train our models but to produce better results, improvements could be made here in future projects.

2.2 Baseline: Synonym Replacement

Our baseline model was a naive model that drew its functionality from the Python NLTK library, specifically its WordNet interface. We used this interface to obtain synonyms for various words in a given input quote, and these synonyms were then inserted back into the quote in the place of the original word. The number of changes made to a quote is a sort of hyperparameter that would need to be changed based on the length of the quote, as we need to balance making enough changes to cause an appreciable change with not making so many changes the quote holds no meaning anymore.

2.3 N-Gram Model

An n-gram model is one solution to the language modeling task. First, define an n-gram as a sequence of n consecutive words. For instance, the phrase "the cat" is a 2-gram, while the phrase "the blue cat" is a 3-gram. In an n-gram model, we make the simplifying assumption that the word x_i depends only on the preceding $n-1$ words. Formally, we are assuming

$$P(x_t \mid x_1, x_2, ..., x_{t-1}) = P(x_t \mid x_{t-n+2}, x_{t-n+3}, ..., x_{t-1})$$

By the definition of conditional probability, we can rewrite this as

$$P(x_t \mid x_{t-n+2}, x_{t-n+3}, ..., x_{t-1}) = \frac{P(x_{t-n+2}, x_{t-n+3}, ..., x_{t-1}, x_t)}{P(x_{t-n+2}, x_{t-n+3}, ..., x_{t-1})}$$

Thus, we are computing the probability of a specific n-gram occurring divided by the probability of that specific (n-1)-gram. occurring. An n-gram model works by estimating the above probabilities through counting the number of occurrences of specific n-grams and (n-1)-grams in a large set of text.

In our particular model, we have n=4; however, it would definitely be interesting to set n to different values in the future, and evaluate those results as well. To generate text, we randomly pick a start word and then allow the n-gram model to run until it hits an END word like a period. Furthermore, the program also replaces any words that appear less than THRESHOLD number of times in the training data with an <UNKNOWN_pos> word. That is, the word is replaced with a word that has the same part of speech as the original word. Whenever an <UNKNOWN_pos> word shows up in the generated sentence, it gets replaced with a randomly chosen word from the training data that has the same part of speech and has also appeared less than THRESHOLD number of times.

The generations using this model are very similar to the results produced by the original Inspirobot. Both are surprisingly grammatical and are often humorously nonsensical. However, people in the survey did not like it. Perhaps we did not make our intentions clear enough; the survey is meant to evaluate how much people *enjoyed* each quote, not necessarily how much

each quote resembles a real inspirational quote. The intent of the original Inspirobot was humor, and we wished to achieve a similar result. But this is difficult, because different people have very different senses of humor, so it's difficult to figure out a good evaluation standard for comedy.

2.4 Recurrent Neural Network

A recurrent neural network (RNN) is another solution to the language modeling problem. What is special about RNNs is that they allow both the values of previous hidden states and the actual inputs to be used when calculating a new iteration of those hidden states. This type of architecture leads to RNNs being able to take into account historical information and process inputs of any length, traits that are useful when trying to predict the next word in a long sequence. For our project, we used the LSTM variation of RNNs to deal with the vanishing gradient problem that is often encountered by traditional RNNs.

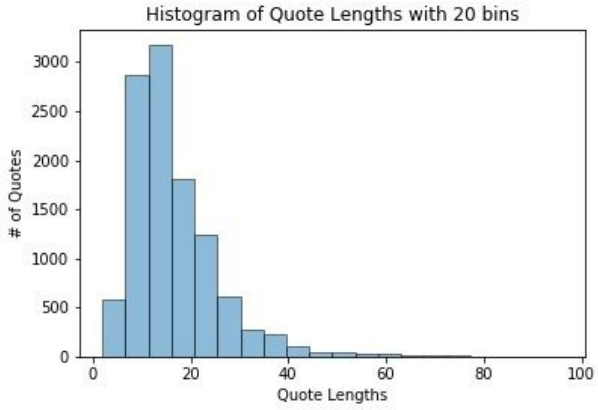
To train our model, we did some brief data preparation. First, we tokenize the text using the Keras Tokenizer class. This creates a vocabulary set that we can use to convert sequences of text into sequences of token indexes. Our model will generate text word by word. Suppose we have the sentence "they are learning data science". Then, we prepare out data by creating pairs of predictors and labels like so:

PREDICTORS	LABEL
they	are
they are	learning
they are learning	data
they are learning data	science

Since our labels are the next word in a sentence, when we give the model sequences of text in the above format, we are training the model to learn this sequence and predict the next possible word.

We now build the neural network with three layers. First, we have an embedding layer with an embedding dimension of 64. Next, we have an LSTM layer with 128 units and 20% dropout. Finally, we have a linear output layer with a softmax activation function. Our model is trained for 50 epochs and then saved.

To generate quotes, we feed our model some random seed word from our vocabulary. Since our vocabulary has no punctuation or stop words, the length of our generated quote has to be determined randomly. To do so, we pick the length of our quote to be some random integer between 5 and 30. We picked this range using a histogram of the quote lengths of the original data set:



From this graph, we see that most of the quotes have a length of 5-30 words.

The generations from our model tend to be quite good with correct grammatical structure and an inspirational sentiment that actually makes sense. Unlike the Inspirobot however, the RNN produces very generic and unhumorous quotes. A major issue with our RNN is that it does not know when to end a sentence and as such, many of the generated quotes stop in the middle of clauses or in other awkward places. Another problem with our RNN is that it seems to use repetitive phrases or is out-right grammatically incorrect. For example, if our RNN is fed the seed words "life" and "success", it produces almost the same quote:

"Life is a journey of a cell enjoying the beauty"

and

"Success is a journey of a cell enjoying the world".

We suspect this kind of issue comes from our data set not having extremely original quotes where words like "life" and "success" can often be interchanged.

The RNN also produced quotes that were just grammatically incorrect. For example, when we feed the word "counselor" into our model, we get the quote

"counselor not a captor sentence for a business".

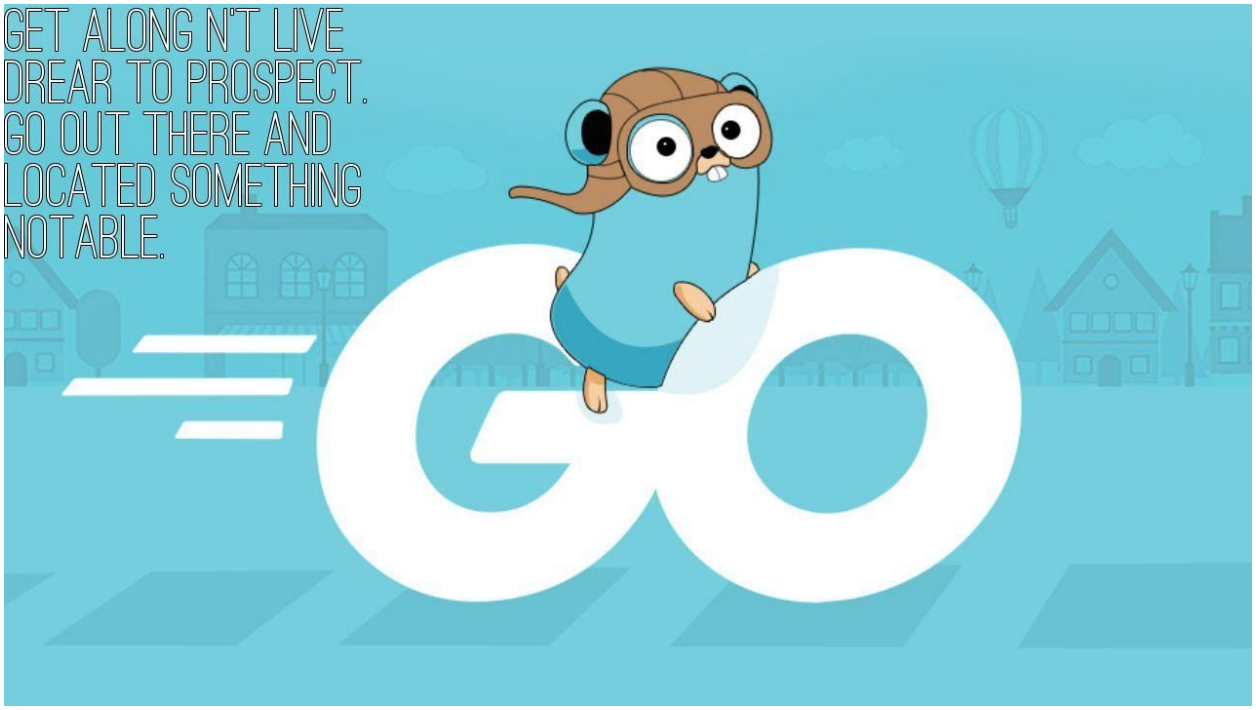
The underlying issue here might be that our model is not precisely fine tuned. The hyperparameters we choose are based on rules of thumb and are more arbitrary choices. Further trial and error or application of research results on lower bounds of the dimensions for word embeddings (https://www.aclweb.org/anthology/I17-2006.pdf) are future improvements we could make. Another possibility is again that the data is flawed and not diverse enough to have words like "counselor" appear often.

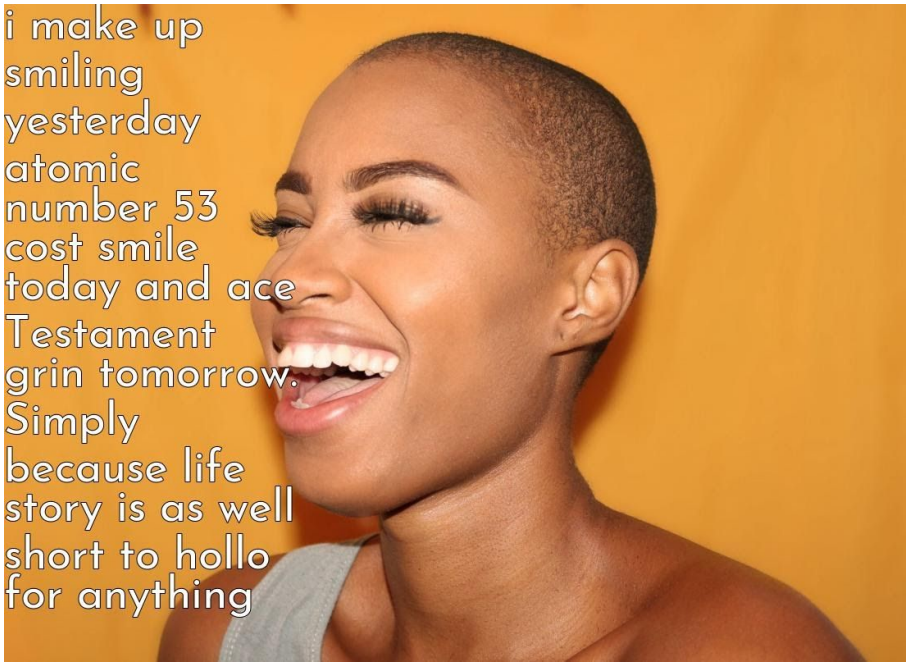
2.5 Creating Posters

To create our posters we wrote a Python script which scrapes Google for a specified number of images related to a specific query. This query was typically either the seed word used for the sentence in the case of the RNN or just a random noun from within the quote. The quote was then placed onto the images in a pseudo random fashion. The best image from the resulting images was then chosen. This was determined subjectively via criteria such as the quote not covering the main focus of the image.

3 Results and Future Improvements

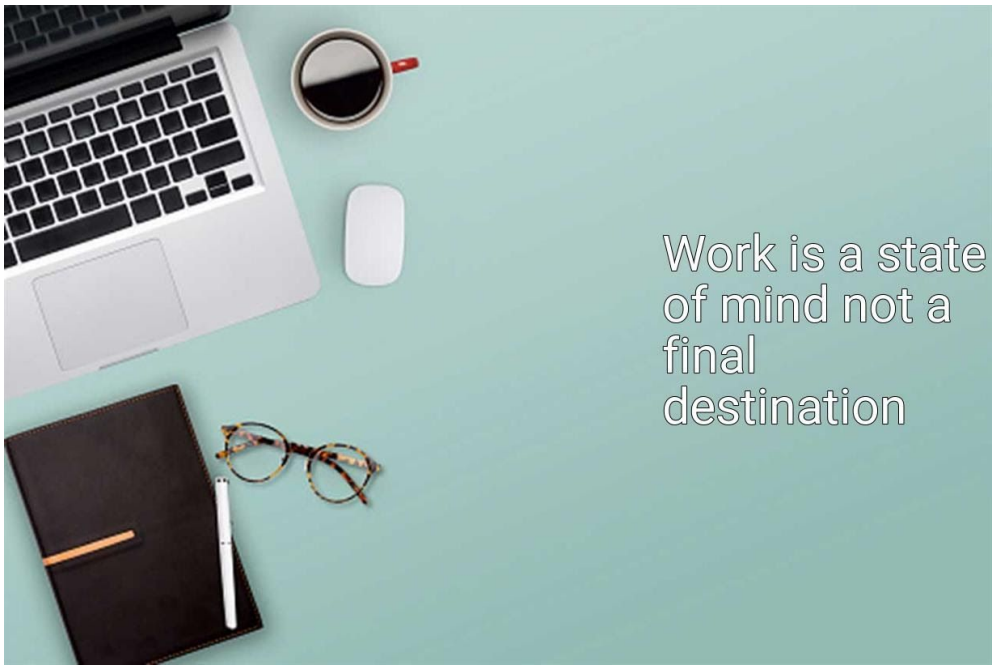
3.1 Baseline Generations




i make up
smiling
yesterday
atomic
number 53
cost smile
today and ace
Testament
grin tomorrow.
Simply
because life
story is as well
short to hollo
for anything

3.2 n-gram Generations

3.3 Recurrent Neural Network Generations

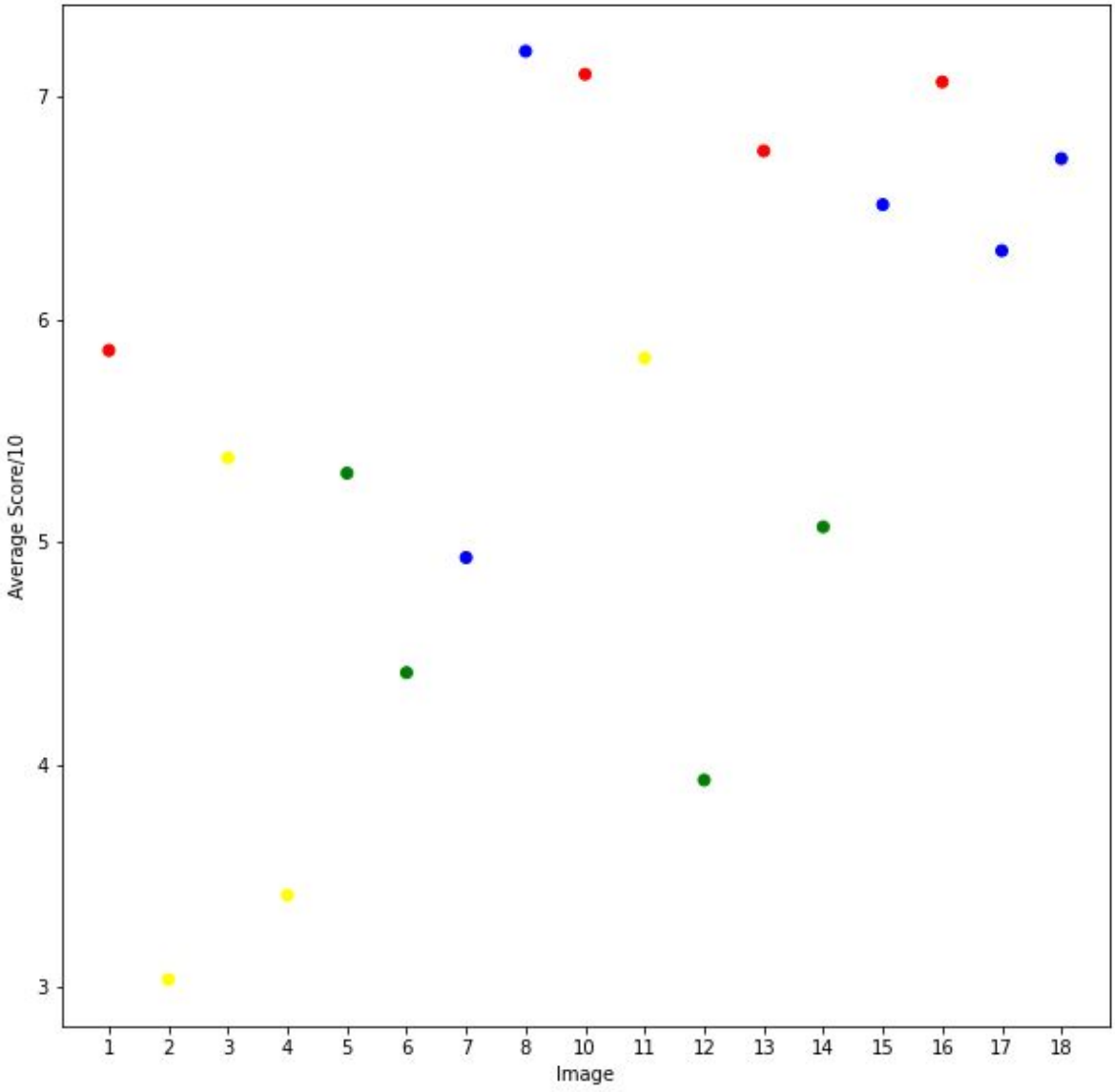



I am not a
teacher i am
not a fighter


TOMORROW IS A NEW
DAY PAINT YOUR LIFE
WITH LOVE

3.4 Error Analysis

To perform some error analysis regarding our generated quotes/posters we created a google survey containing the posters and asked users to rate the images on a scale from 1-10. As we can see from the figure below, the images with real quotes and the quotes created via RNN were rated higher than the images with quotes created via the baseline model and the n-gram model. N-grams here wildly underperformed despite the fact that we thought those quotes (see above) were the funniest and most enjoyable. One thing that might have helped here is providing some more context to Inspirobot and what we were trying to create. Some other reasons that the n-gram quotes could have underperformed was that they are not as aesthetically pleasing posters compared to the RNN. Compared to RNN generations, we see above that the n-gram generations are harder to read and much longer, possibly leading some people to enjoy them less.



3.5 Future Improvements

We think one of the major improvements we could make is finding better data. Namely, we want a data set that is much larger and more thoroughly vetted. Preferably, this data set would be more consistent in the languages it used and be more formal, avoiding issues with weird slang or internet-speak.

We could figure out better evaluation methods, as well as implement more part-of-speech tagging since the program often made grammatical errors.

For the RNN specifically, we think that using a character RNN, that is predicting the next character in a sequence, would get around the issue of knowing when to end a sentence. There are some drawbacks to using character RNNs however as these models must deal with a much longer history and can often produce gibberish words.

4 Who Did What?

Mike worked on the baseline model, the code for generating the images with a quote, the survey, and the paper.

Kyle worked on the data cleaning, the code for the RNN model, the slides and the paper.

Nicole worked on the code for the n-gram model and the slides.

5 References

[1] https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture06-rnnlm.pdf

[2] https://courses.grainger.illinois.edu/cs447/fa2020/Slides/Lecture11.pdf

[3] Olah, C. (2015, August 27). Understanding LSTM Networks. Retrieved from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] Patel, K., & Bhattacharyya, P. (2017). Towards Lower Bounds on Number of Dimensions for Word Embeddings. Retrieved from https://www.aclweb.org/anthology/I17-2006/