Cross-Topic Quotes Generation using Word Embeddings (QuoteGen)

Rohak Singhal 3035242475
Ish Handa 3035238565
Code: https://github.com/krohak/QuoteGen

1. Abstract

The aim of this project is to apply modern day Machine Learning techniques to Generate Quotes across different topics (Cross-Topic Quotes Generation). In this paper we list the various methods we have explored to generate the quotes as well as propose a Sequential, Word-Embeddings based model to achieve the most coherent results.

2. Introduction

Quotes are considered to be wise echoes of people, often recited by everyone. We apply novel Machine Learning techniques to not only create new quotes, but also make these quotes cross-topic.

Our generator creates quotes differently from other pre existing generators on the internet as we generate the quote according to the categories input by the user. The user is required to enter topics from a list of categories provided by us- The first topic is the seed to the system which is used to complete the final quotes using other categories. Our focus was on retaining the essence of the topics provided, so that when quotes from these topics are combined, the generated quote is interesting and amusing.

The model we propose uses word-embeddings to map the words in a higher dimensional space. This ensures that words with similar meanings get mapped close to each other. It then uses Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) to understand the context of the sequences of words it receives. Finally, to generate and output a cross-topic quote, we sequentially combine 10 models trained on quotes each with a different category.

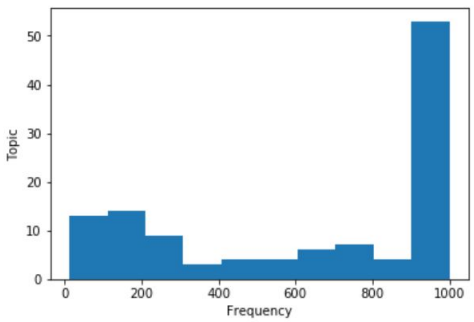
2. Related Work

Quote generators are quite popular since we found that a few of them exist online. Despite that, not a lot of them are implemented using Machine Learning techniques; there are some projects on Github which use Markov Chains to generate quotes and a few which use Neural Networks. However, none of them involve cross-topic generation of quotes using user selected categories.

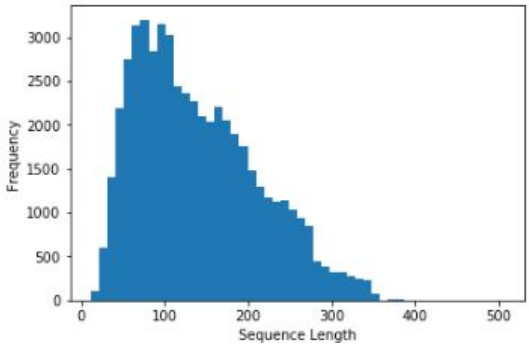
3. Dataset, Features and Preprocessing

Because deep learning algorithms require a large amount of training data for the model to perform well, we chose a CSV dataset of 76,000 quotes available on Research Gate by Madadipouya, K.

3.1 Data Filtering and Cleaning



The Dataset has a total of 76,000 quotes, each labeled with their respective topic and author. Around 110 topics exist in the data, however some topics had less than a 900 quotes (as seen from the graph) and therefore we decided to filter them out of our data. Upon removal of the limited categories, we were left 53 categories. Since some of them were very closely related and our task required training multiple models, we decided to handpick 10 categories from the 53, which seemed distinct. We were then left with 10 categories, each containing around 1000 quotes, making a total of around 10,000 quotes.



The figure above demonstrates the distribution of words amongst the quotes. We can see that most quotes contain about 100 words.

Once we had our base dataset to work on, we cleaned all our quotes by converting them to lower case and removing other unnecessary punctuations while adding a space between special characters like ‘ . ’ and ‘ , ’ which we wanted to retain.

3.2 Preprocessing

Since we implemented multiple methods as explained in Section 4, we had to experiment with various preprocessing methods.

3.2.1 Bag of Words

Our first attempt involved the use of Bag of Words approach. This involved mapping each character to an integer value using a Dictionary. With the help of this Dictionary, we created a matrix of shape 10,000 x 200 (where 10000 is the row representing number of quotes and 200 is the word limit chosen by us). TAny quote longer than 200 words will get concatenated, which made the dimension of the matrix uniform. This new matrix had quotes in their ID form, i.e. converted to a sequence of integers, which can be fed into the character based model.

For switching to the words based model, we updated the dictionary to map each word to an integer rather than a character to an integer. This lead to the creation of an IDs matrix with around 15,000 unique words, and the same 10,000x200 shape as before.

3.2.2 GloVe Word Embeddings

After experimenting with the Bag of words approach, we decided to utilize Word embeddings using Stanford’s pretrained GloVe: Global Vectors for Word Representation. The GloVe algorithm utilizes the count data while simultaneously capturing the meaningful linear substructures (Jeffrey Pennington, Richard Socher, and Christopher D. Manning). This means that words with similar meaning lie close to each other in the vector space.

As the GloVe algorithm revolves around the concept of analogy preservation (Mikolov et al., 2013), it leads to an improvement in both accuracy and interpretability as compared to Bag of Words.

To achieve the above, we tokenized each word in our final dataset of 10,000 quotes using the keras tokenizer. For vectorizing each word, we used GloVe 100 dimensional word embeddings pre-trained on Wikipedia and Gigaword datasets.

3.2.3 Sequences and Next Sequences

We generated sequences and corresponding next sequences out of the quotes, in which we create a sequence of 100 character / words and their corresponding next character / word. We do this for all the words present in the ID matrix of a category.

4. Methods

We developed multiple LSTM models to generate quotes, given seeds and categories. These models are discussed in section 4.1, 4.2 and 4.3. All the three models involved the Sequential Generation method (section 4.4) to handle the Quotes Generation.

4.1 Character Level Model

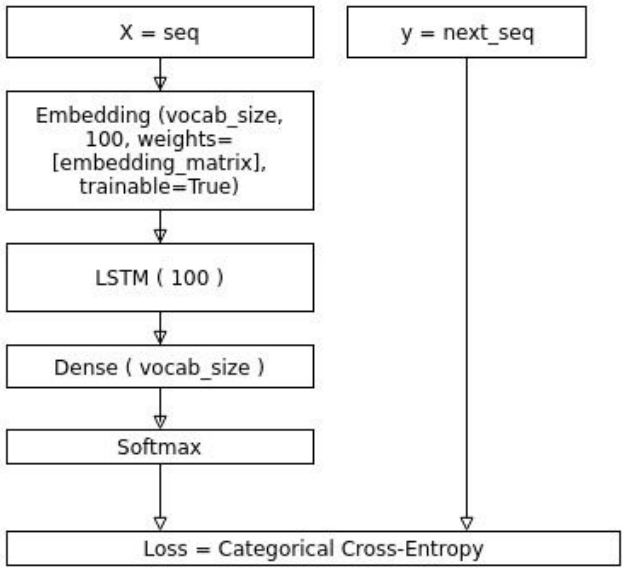
The character level model takes input sequences of characters and predicts the probability of the next occuring character. An LSTM of size 256 with dropout 0.2 was added to the model. The output layer is a Dense layer with Softmax, the same size as the dictionary to predict the probability distribution of each character in the dictionary.

4.2 Word Level Model

We experimented with a model similar to the character level model described above, using the dictionary of words in the dataset instead of characters. We also created another model with non one-hot encoded outputs, which had Relu as the activation of the final Dense layer with a single output.

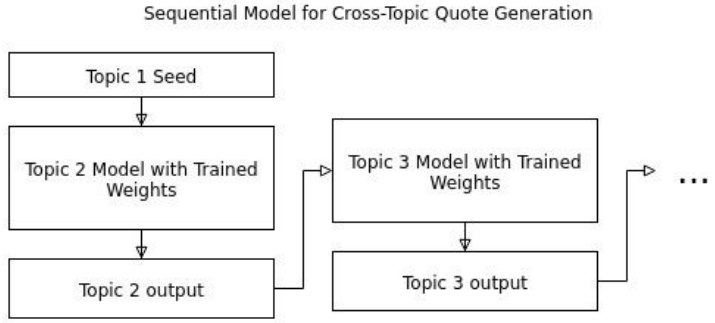
4.3. Word Embeddings Based Model

For the embeddings based model, we used GloVe vectors in 100 dimensional space. LSTM hidden layers with 100 memory cells and a Dense layer the size of the vocab was added to the model. The output of the Dense layer is a vector of probabilities for each word in the dictionary. We then take the maximum of these probabilities to output the next word, given a sequence. Categorical Cross-Entropy is used to measure the error since the output is one-hot encoded. We use Adam to perform SGD.



Since we train each neural network model on a particular category category of quotes, we trained 10 models for the 10 selected categories.

4.4 Sequential Generator



These 10 trained models are then sequentially combined as per the figure above. A seed is chosen randomly from the quotes of the first topic and input into the trained model of the second topic. We have obtained results combining two categories, but this idea can be extended to as many categories as demonstrated above.

5. Experiments

5.1 LSTM and Dense Layer Numbers and Sizes

A single LSTM layer of size 100 outperformed 2 LSTM layers of the same size in accuracy. Increasing the

number of dense layers did not have any effect on the accuracy of the model in training.

5.2 Embedding Layer with preloaded weights and training vs non training mode

We also experimented with the embedding layer with preloaded GloVe weights vs non-preloaded weights. The pre-loaded weights achieved a higher accuracy faster. Similarly, if we left the weights to train further, we achieved a higher accuracy in lesser number of epochs.

5.3 Batch Sizes and Epochs

A batch size of 24 and 30 epochs were chosen to train after experimenting with these parameters. This provided a good trade-off between training time and accuracy. An accuracy of around 50% was achieved after training for 30 epochs with batch size 24.

5.4 Temperature and Randomness of prediction

After including a log-exponent temperature function and randomly sampling from the multinomial distribution of predicted probabilities, we found that the generated quotes were highly incoherent. Thus, we decided to not include temperature and randomness.

6. Results

6.1 Word Level and Character Level Based Generator

Char Level Model Output
out to toeet to pelilion the pelsle ane the pasty le a eeel to iet intolved in politics is that it so be toeethong th teet to the of the peotle the semsblicans are the politics...

Word Level Model Output
thats thats thats ancients thats thats thats thats families thats ancie 9k thats thats thats thats thats thats thats thats thats thats ancients thats thats th thats thats thats ...

Higher Temperature on Word Level Output
quantities equating neocon periods comfortably spot acquaintances slamming 9k streets endless compete thoughts inalienable sea thoughts vinyl libertarian ginsbergs...

Above are a few outputs from the initial models we explored. The outputs are often incomprehensible and repetitive since the models cannot extrapolate the given input sequences.

6.2 Word Embedding based Generator

death + family :
----- Input seed: *death , so called*
----- Output: *the caring adult in the family*

funny+science:
----- Input seed: *being funny with a funny voice is*
----- Output: *not rocket science*

love + death:
----- Input seed: *you can only embrace it when*
----- Output: *you dont want to live*

We achieved a higher performance using GloVe word embeddings as demonstrated above. The generated quotes are more coherent, retain the essence of the topics involved in its generation and are often quirky and humorous.

The property of GloVe to map words with similar meanings to points nearby on the vector space proved to be ideal for our generator. Since some words are specific only to certain topics of quotes, models which have not experienced those words can still get an idea about their meaning since the model may have been trained on words / sequences similar in meaning.

6.3 Human Evaluation

To evaluate the success of our generator, we received help from 10 people, who tried our generator.
Each user was asked to rate the generated Quote on a scale of 10, with the following criteria in mind; Grammar, Meaning and Degree of relation to their chosen categories.

After compiling all the results, our generator seemed to have performed reasonably well with an average score of 7 stating that the degree to which the quote was related to the categories was quite high.

7. Conclusion and Future Work

In this project, we proposed a Sequential, Word Embeddings based model to generate quotes, given a seed and up to two categories.

Our main goal was to produce an interesting and humorous new quote, based on given topics. We also wanted to retain the essence of the topics so that the generated quote is a perfect mix of the chosen topics. As we can see in the results above, these goals have been achieved through the Sequential, Word Embeddings based model which we trained.

In the future, we would like to perform cluster analysis on the quotes dataset to find out similar topic clusters that are distinct from each other instead of using the hand picked ones. This could lead to an increase in the performance of our model, allowing the user a bigger and better list of categories to choose from instead of generating quotes amongst similar categories. We would also like to extend this idea to incorporate more than 2 categories and even take an input seed from the user for the system to complete the quote based on a chosen topic. This is currently work in progress at the time of writing this report, and will be updated on our GitHub repository.

Note: An ipynb file explaining the code and the scripts used for replicating the results above can be found in our repository (link on the first page heading)

8. Contribution

Rohak Singhal: Methods

Ish Handa: Preprocessing

9. References

1. Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Distributed representations of words and phrases and their compositionality. CoRR, abs/1310.4546, 2013. URL http://arxiv.org/abs/1310.4546.

2. Madadipouya, Kasra. CSV dataset of 76,000 quotes, suitable for quotes recommender systems or other analysis. 2016/07/03. URL https://www.researchgate.net/publication/304742521_CSV_dataset_of_76000_quotes_suitable_for_quotes_recommender_systems_or_other_analysis

3. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. URL https://nlp.stanford.edu/pubs/glove.pdf