**Final Report – Yelp Review Text Generation**

Yue Zeng (yz723), Haoyun Chen (hc321)

1. Introduction

This project focuses on text generation (language modeling). We chose Yelp reviews as our corpus and wish to generate positive reviews for restaurants, markets, etc. This task can be applied to real life. The challenges mainly come from 2 perspectives: 1) Yelp reviews include many initials/shortcuts/slangs such that the corpus contains many rare words; 2) Yelp reviews contain many uncommon punctuation combinations, so the raw data need to be cleaned with care at the beginning.

To accomplish this project, we applied both Long short-term memory (LSTM) and Markov Chain (MM) models, which are discriminative and generative respectively. We first preprocessed the text in the corpus. Then we trained both models based on the text and used the models to generate texts. In addition, we analyzed and evaluated the results quantitatively and qualitatively. Finally, we discussed the pros and cons of the two methods from multiple perspectives.

1. Method
2. Text Preprocessing

We first loaded the CSV review data file using the ‘pandas’ module. Because we considered that longer reviews are containing more information and are more guaranteed with higher quality, we used the ‘pandas’ module to filter out the short reviews (below 30 characters). We only reserved reviews with rates of 5 to compose the positive reviews dataset. And we applied the ‘re’ module and regular expressions to remove special characters such as brackets and tabs from the raw data to prevent overfitting. After that, we used the tokenizer functions from spacy to tokenize all the review texts. We separated our dataset into the training set and testing set. The training dataset contains 2000 reviews, and we used 10 reviews to test our models.

1. LSTM Model

The LSTM network is applied because we wanted to keep the longer term of memories based on the relatively longer text input. The structure consists of multiple layers of neurons, with each layer containing a long short-term memory (LSTM) unit. The LSTM unit is composed of a forget gate, an input gate, an output gate, and a memory cell. The forget gate is responsible for discarding irrelevant information from the memory, the input gate is responsible for controlling the flow of information into the memory, the output gate is responsible for controlling the flow of information out of the memory, and the memory cell is responsible for storing information.

We wished the users could input a complete sentence to the model to generate texts. So we made the model take 15 vectors of tokenized words as the input. The words are one-hot encoded at the input such that each input vector is having a dimension of the size of the vocabulary in the corpus. We applied an embedding layer to handle the input to transform them into vectors with lower dimensions as large as the length of the input sequence. In the following of each layer, the numbers of neurons were all decided to be the multiple of the length of the input sequence, (typically 6\*15,) which empirically achieved better effects. We added 2 layers of LSTM layer and 2 dense layers in the neural network and used the SoftMax function to generate the output of the predicted probability distribution of the next word. We used categorical cross entropy as the loss function and the Adam optimizer as the stochastic gradient descent algorithm. We trained the model for 300 epochs.

1. Markov Chain Model

Parameter inference in Markov chain text generation models involves estimating the parameters of the transition probabilities of the Markov chain. This is done by using a training corpus which is our Yelp review texts. The transition probabilities are the probability of transitioning from one state to another, e.g., the probability of transitioning from the word 'the' to the word 'restaurant' in a sentence. The parameters are then estimated by counting the number of observed transitions in the training corpus and dividing them by the total number of transitions. Once the parameters have been estimated, they can be used to generate new text samples by following the transition probabilities.

Our project applied the 4-gram model which was predicted based on 3 words transition. Therefore, the transition matrix needed to keep the transition frequency between 4 words. We used the ‘markovify’ [1] module in Python to fulfill this task.

1. Result

After we trained both models, we applied them to generate reviews based first on real data, then on synthetic data generated by the Markov chain model. We used the first 15 words of the test texts as the seed texts for our models and generated the same length of words (e.g., 200 characters) for both models each time.

Then We calculated Bleu scores of the generated texts to quantitatively evaluate the quality of text generation. It compares the prediction texts against one or more reference texts and assigns a score based on how closely the predictions match the reference texts.

The table below shows the quantitative performance including the Bleu Score and precision of the 2 models :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bleu Metrics on Real Data | | | | | |
| Models | Bleu Score | Unigram Prec. | 2-gram Prec. | 3-gram Prec. | 4-gram Prec. |
| Markov Chain | 5.17 | 41.86 | 15.48 | 11.95 | 9.75 |
| LSTM | 0.30 | 36.64 | 3.07 | 0.12 | 0.06 |
| Bleu Metrics on synthetic Data | | | | | |
| Models | Bleu Score | Unigram Prec. | 2-gram Prec. | 3-gram Prec. | 4-gram Prec. |
| Markov Chain | 0.22-5.86 | 40.75 | 4.80 | 0.98 | 0.25 |
| LSTM | 0.28 | 44.01 | 7.08 | 1.21 | 0.25 |

For the real data, we used 10 real Yelp reviews not appeared in the training set as the test. In this situation, we found the Bleu score of the Markov Chain model is much higher. However, one downside of our Markov model approach is that the ‘Markovify’ module could not handle the input not discovered/fitted in the training process. So for some ‘unknown’ feeding texts, this method would output errors. Therefore, the test set does not always work for the module.

For the synthetic data, the Bleu score of the Markov Chain model varies greatly from different generated testing texts, while the score of LSTM is relatively stable.

1. Discussion

From the test result, we conclude that from both sources of data, the Markov model performs better in most cases, despite the issue that it would not generate output when occurred with unvisited words in the real test data.

Also, from the perspective of semantics, we found the Markov model generated better results than our neural network because its generated text looks more like real-world reviews written by human beings. We ascribed this to that the texts in the corpus were all having similar topics, and it would be more probable that fixed phrases and terms are used more frequently. Those patterns would take up large weights in the Markov models such that the many predictions of such phrases were in accord with the real data.

However, the Bleu score of the Markov model on synthetic data fluctuated and sometimes are lower than that of LSTM. Because every time both the reference text and the prediction texts are generated randomly according to a probability distribution, it is probable for the model generates rare words or combinations which made the sentence deviate from the reference.

Our LSTM model usually performs not as well as the Markov model both in the sense of the score and coherence. For both the real-world data and the synthetic data, LSTM couldn’t generate a sentence that makes sense. Because of the limitation of our hardware, we chose to apply a relatively small model and trained for a short time. For the model itself, we trained it in 300 epochs which cost 15 hours. We believe the model’s performance would improve a lot if we doubled our epochs, enlarge the training model, and feed the model with more reviews. The advantage of our LSTM model is that its performance was much more stable than that of the Markov chain for both real and synthetic data. We assumed that this was due to LSTM’s ability to remember long-term dependencies. As a result, LSTM’s predictions were based on longer input and their higher-order patterns, which provided robustness for its output. Also, unlike the Markov model, LSTM always generated the same output given fixed input, which made its behavior more predictable.

We have also compared the pros and cons of our models from different perspectives. Pros of the LSTM Model include:

* Quality/Correctness: LSTM models are capable of learning to capture long-term dependencies from a sequence of text, thus providing more stable accuracy.
* Interpretability: Some methods such as visualizing the weights of the model can be used to gain insight into how the model works.

Pros of the Markov Chain Model include:

* Quality/Correctness: Markov chains can produce results that are close to the real-world distribution of words and phrases, which can lead to more accurate and realistic text.
* Data, Time, and Computational Requirements: Markov chains require very little data to train and are very fast to compute.
* Interpretability: Markov chains are very interpretable as the underlying structure of the model is simple and easy to understand.

Cons of the LSTM Model include:

* Quality/Correctness: LSTMs require a large amount of data to be effective, and their results can be inconsistent when trained on small datasets.
* Data, Time, and Computational Requirements: LSTMs require a large amount of computing power, and they can be slow to train due to their complexity. As LSTMs are trained using backpropagation and stochastic gradient descents, they require a large amount of time and data to have better performance. And a smaller model and low train time will lead to poor results.
* Interpretability: Due to their complexity, LSTMs can be difficult to interpret.

Cons of the Markov Chain Model include:

* Quality/Correctness: Markov chains are limited in their ability to capture long-term dependencies, and they can produce results with lower accuracy occasionally.