Comments Without Food: Generating Fake Restaurant Reviews Using RNN

Tianxiao Hu, Yi-Fang Lee, Zhenbang Chen, Wenyuan Gao
{tianxiao.hu, yifang_lee, zhch, wenyuan_gao}@berkeley.edu

May 13, 2019

Abstract

Recurrent neural networks (RNN) can be used to learn sequential data and generate similar data. In this project, we focus on restaurant review generation using RNN. The model learns from restaurant reviews on *Yelp.com* (which is wrote by human). Our goal is to generate reviews realistic enough to fool native speakers.

We followed two distinct branches in pursuing this goal: character-level language model and word-level language model. For character-level models: we took vanilla two-layer stacked LSTM network as our baseline. Then we improved the network structure with additional embedding layer and attention layer. For our word-level model, it starts with pre-trained word2vec embedding layer, along with two stacked GRU layers. Also, dropout is used to prevent overfitting. Finally, we compared our model with the state-of-the-art GPT-2 language generation model (simplified version).

We used the *Yelp Open Dataset* to generate fake reviews. The dataset contains 3+ million 1-5 star restaurant reviews. For each network architecture, we trained it on two datasets: 1-star reviews and 5-star reviews for better comparison. Our results show that even two-layer RNN can generate fake reviews that are hard to tell whether written by a human.

1 Problem Statement and Background

User reviews are commonly used to provide rich information for both customers and companies. The review contents provide information about product quality, services, and price. Customers tend to spend their money and time on restaurants with positive reviews and high ratings. Companies like Yelp or Amazon create recommendation list for customers based on the user-written reviews and the content may also affect the revenue of the product or service provider. Due to the huge impact of reviews, some companies started hiring crowd-turfing agents to manually generate positive reviews [1].

With the improvement of deep learning and natural language processing techniques, automatically generating reviews for restaurants gradually became a trend and it is also a challenging topic since more and more researchers are focusing on how to detect this kind of machine-generated reviews [2]. Some companies also started using review filters to keep the reality of user comments [3]. Therefore, a fake review should provide specific information about the restaurant and the English sentences should be natural enough to fool native speakers. How to process existing content and let the neural network to achieve this goal is a challenging and interesting topic in natural language processing (NLP) field. Also, there are not so many articles comparing different NLP models on review generating. Thus, we decided to choose generating efficient fake reviews as our project topic.

Since **Yelp.com** publishes their reviews about businesses for academic usage, we decided to use its dataset as our training data [4]. The dataset contains more than 3 million reviews for business with ratings from 1 to 5 stars. Despite star rating, the reviews has corresponding business information, such as attributes and categories. We specifically focused on restaurant reviews in our project to train our different models. These models are trained until loss converged or achieved certain threshold, then the generated results are picked up manually.

2 Methods and Experimental Results

2.1 Data Preparation

We downloaded our dataset from *Yelp Open Dataset*[4]. After extracting, all data are stored in JSON format.

In data preparation, we used 2 files in the dataset folder: *review.json* and *business.json*. First, we converted them to CSV format. Then we only kept columns in *business.csv* which have "Restaurant" tag in their categories. Next, we merged reviews and restaurants by key "business_id" to filter only restaurants reviews.

During the process, we noticed that some characters in other languages (Chinese, Korean, Japanese...) also appear in the review text. We chose to remove these characters, keeping only English characters in reviews. We also removed line break symbols so that each review will hold a single line.

For each model, we trained it on two datasets separately. One model was trained on 1-star reviews and another on 5-star reviews. The key "stars" can be used to generate different input datasets.

2.2 Model Overview

We followed two distinct branches to generate reviews: character-level language model and word-level language model. For character-level language model, the minimum distinguishable symbol is a character, which means we need to feed multiple characters to our network and let it predict the next character.Similarly, for word-level language model, a word is the minimum distinguishable symbol. Our network will learn to predict the next word according to former words.

There is an example for better comparison. For a single sentence "I like the food.", character-level language model will separate it as "I␣␣␣l␣i␣k␣e␣␣␣t␣h␣e␣␣␣f␣o␣o␣d␣." while in word-level language model as "i␣like␣the␣food␣."

There are some pros and cons for character-level language model:

- Pros:
  - The vocabulary size is relatively small. This makes it possible to train on a large corpus.
  - It's easy to setup: doesn't need too much data preprocessing.
- Cons:
  - The model can end up generating very long sentences. Because the punctuation appears less frequently during training process.
  - It's not as good as word-level language models at capturing long-range dependencies. The earlier parts of the sentence may not affect the later part of the sentence.
  - Compared to word-level language models, the character-level language model is more computationally expensive to train.

We began with a simple character-level model, which has two LSTM layers with 256 cells each followed by a feed-forward layer. Then we added extra embedding and attention layer on top of the vanilla character-level RNN to seek performance improvement. Afterwards, we switched to word-level language model. It employs Gated Recurrent Units (GRU) [5] with dropout and uses pretrained Word2vec [6] weights to initialize the embedding layer. We also compared the word-level model with the state-of-the-art GPT-2 model. Due to computational resource limits, we used the simplified version. The simplified GPT-2 model has 12 repeated transformer blocks with multi-head attention structure. An overview of network structures can be found in Figure 1.

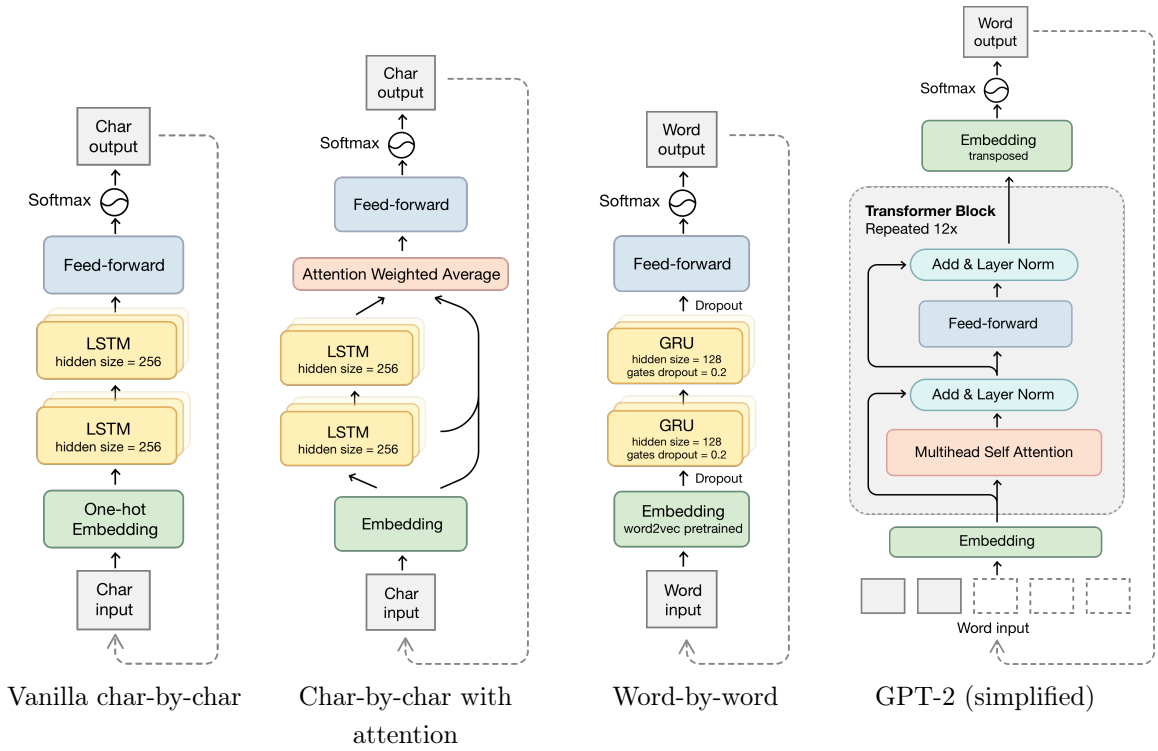

Figure 1: Structure overview of four models experimented. Left to right: Vanilla char-by-char, Char-by-char with attention, Word-by-word, and GPT-2 (simplified) correspondingly.

Vanilla two-layer LSTM.

Our baseline model is quite straightforward. The model begins with a one-hot embedding layer, which will convert characters to sparse vectors. Then followed is two stacked LSTM layer, each having 256 cells. A softmax layer is deployed to generate the convert numerical value to possibility.

We extracted 20, 000 reviews as the input dataset and trained our model using RMSprop optimizer. The learning rate started with 1e-3, then would decay when the model encountered a plateau. Cross-entropy error was used to evaluate the model performance. For 1-star model, it was trained 24 epochs and the loss dropped to 0.99. For 5-star model, it was trained 25 epochs with the final loss being 1.01. The batchsize is 1024 and maximum 60 characters will be fed to the network for predicting the next character.

LSTM with Attention Layer.

As an improved version of vanilla two-layer LSTM, we chose to use *textgenrnn*[1] package, a Python 3 module on top of *Keras*/*TensorFlow*, to create attention-based char-by-char level model. As displayed in the Figure 1, the general structure is similar to the vanilla one, but an attention layer is applied after stacking two 256 cells LSTM layers. In order to compare the result easily, the training dataset is the same as the vanilla LSTM with 20,000 reviews for each 1-star and

[1]https://github.com/minimaxir/textgenrnn

5-star category. In order to create sensible text consistently, the recommended training loss for *textgenrnn* model is less than 1.0, so we trained 5-star reviews with 18 epochs and 1-star reviews with 22 epochs to achieve both loss value around 0.973. For parameters, embedding size is 100, batch size is 1024, and 30 tokens are considered before predicting the next one.

Word-level Generation.

We also experimented with word-level RNN. The model consists of two stacked GRUs and a word embedding layer (pre-trained with Word2vec). Compared to char-level models, more parameters are needed in the word-level model, which makes word-level models more difficult to train. We limited the vocabulary size to about 8000 with respect to frequency. Words with lower frequencies were pruned.

We also pre-trained a Word2vec word embedding model on the same dataset. The weights generated are used to initialize the word embedding layer to map each token into the latent space. We conducted an experiment and found that using pre-trained Word2vec weights can significantly speed up the training process, which supports our assumption that training the embedding and feed-forward layers are a large cost since they share 80% of the parameters. We will talk about the experiment in the next section.

For models, we chose to use GRU over LSTM in word-level models. There are only two gates (the update gate and the reset gate) in GRUs. Ergo, GRUs are computationally cheaper compared to LSTMs. Given that GRUs perform comparably to LSTMs [7][8], GRU is used in our word-level model. We employed dropout rates i) of 0.5 between layers and ii) of 0.2 after gated outputs within GRU.

Teacher forcing [9] has been proved an effective technique when training recurrent models. At each training step, the correct label instead of the current recurrent output is fed as the input for the next step. We implemented this with the *TimeDistributed* module in *Keras*.

GPT-2.

Finally, we compared our model with the state-of-art model GPT-2. GPT-2 is a transformer-based language generation model, which is proposed by OpenAI in 2019 [10]. In the experiment, we use the simplified GPT-2[2], which contains 12 repeated transformer blocks with multi-head attention structure. We only fine-tuned the GPT-2 model using 1,000 reviews, which is enough to generate realistic comments. From the experimental result, we can see that the performance of our model (Word-level Generation) is almost close to GPT-2, which proves the accuracy and reliability of our model.

2.3 Generation Results

We mainly used *TensorFlow* and *Keras* deep learning framework in our implementation. Code is hosted on https://github.com/TianxiaoHu/FakeYelpReview. To enable GPU acceleration, we ran our experiments on Google CoLab and AWS instance.

We compared the generation results of the four model structures. The 1-star models and 5-star models are trained separately on corresponding review data sets extracted from *Yelp Open Dataset* with respect to the number of stars. Generation result samples are listed in Table 1 and Table 2.

3 Analysis and Conclusions

3.1 Character-level Language Model

In the experiment, we began with the character-level language model. The character-level models are generative models, which are supposed to generate very long sentences. We set vanilla char-

[2]https://github.com/minimaxir/gpt-2-simple

Table 1: One-star review samples by different model

Model	1-star reviews generated
Vanilla char-by-char	Will not come here again. We were disappointed that I was the only ones that the hostess stopped here. Because of the Kateron that there was a bad service and the food was bland and overcooked.
Char-by-char with attention	The food was okay. The rice was ok, but the waitress was pretty stupid for the first time. We were told they were out of bread. The sandwich was so tough they didn't have a couple of salads.
Word-by-word	If I could give this place 0 stars I would. I ordered a pepperoni salad and was told it was a small portion of meat. I was ordering it. I said it was supposed to be cooked. Also, I got a mouthful of chicken on the side and he was over cooked.
GPT-2 (simplified)	Worst dim sum place that I have ever been. They have black hair in their food and urine. The hostess had the whole restaurant and all of its personnel in front of her at the same time. It was incredibly rude and graphiced everything we did and where we were served.

Table 2: Five-star review samples by different model

Model	5-star reviews generated
Vanilla char-by-char	This is the best in Phoenix area! The service was amazing. It was nice and tasty and creative in a large group of friendly and attentive staff and no other omelet over the exception. I also order the potato, chicken and the chicken parmbialese.
Char-by-char with attention	I love this place. I have been going to this location and was very convenient. The food was amazing and the service is always good. The owner is super friendly and helpful. The service was fast.
Word-by-word	It was amazing. I had to try the el balls, and the seasoning was delicious. The amount of meat which was amazing and the sauce was the best, I have ever had. I will definitely be back. Biloxi fried ice cream and the fried banana. Both are delicious.
GPT-2 (simplified)	We like to think that this is the best Thai restaurant in the area and possibly in Mississauga as far as Thai is concerned. The food is certainly well prepared and the staff is extremely friendly. The ambience is calming and inviting. A traveler in need of a table? This is a must try!

by-char network as our baseline, which uses one-hot embedding and two-layer stacked LSTM network followed by a feed-forward layer. Then we improved the vanilla network by adding an extra embedding layer to the input and an attention layer ahead of the feed-forward layer. The additional embedding layer can better encode the character into meaningful vectors than the one-hot embedding, while the attention layer next to the stacked LSTM layers can assign different weights to the LSTM cells and propagate more accurate character parameters to the feed-forward layers, which can improve the model performance to some extent.

From the reviews generated by vanilla two-layer LSTM model and our improved network, we can see these two models generate basically similar reviews in syntax and sentence pattern, although the reviews from the improved network are a little more complete and readable due to the benefit of the additional embedding layer and the attention layer. However, since character-level models only focus on the characters without the consideration of entire words, the character-level models have great limitations and reviews generated from both of the two models are not natural enough and incoherent in semantic. To get rid of this limitation, we switched to word-level models, which can consider the semantic relations of words in sentences as a whole and thus can generate more consecutive reviews.

3.2 Word-level Language Model

During training process, the word-level model considers each word as a minimal token, while character-level models treats each character as a minimal token. The two levels of text generation models basically has very similar model structure, except for the different granularity of the input and output tokens. Apart from the minimum distinguishable symbol, the difference between word-level and character-level models mainly lies in the embedding layers. Since the input sequence of character-level model is greater, it requires longer training and generating time. Also, the stacked RNN structure of character-level model is more likely to have gradient vanish or explode problem with the increasing length of the input sequence.

For the word embedding layer, we basically tested two kinds of embedding initialization in our experiments, that is to set the embedding layer by random initialization or pretrained Word2vec weights. The curve of training samples and loss is displayed in 2, we can find that using pre-trained Word2vec embedding significantly improved the training speed, and the performance was slightly better, compared to training tabula rasa. All other settings remained the same, both using Adam [11] as the optimization algorithm, and 1e-3 as learning rate.

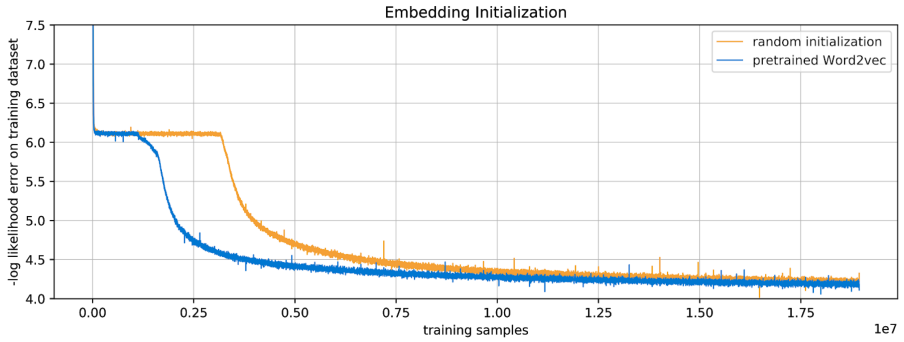

Figure 2: Comparison of training with versus without pre-trained Word2Vec embeddings

The word-level model we implemented is based on word embedding and GRU layers with dropout. As we discussed before, the word embedding layer initialized by pre-trained Word2vec can converge more quickly, so we implemented the model which starts with a pre-trained Word2vec embedding layer, then followed by two stacked GRU layers and a feed-forward layer. The baseline

model we selected for our word-level model is GPT-2, which is the state-of-the-art text generation model. Different from RNN-based models, GPT-2 is based on transformers, containing 12 repeated transformers blocks with multi-head attention structure.

The reviews generated by the two word-level models are more coherent in semantic and sentence pattern compared to character-level models. We can see some long and complex clauses appears in the generated reviews, which make these reviews look more like written by real humans. To compare these two word-level models, we can notice that the reviews generated by our RNN-based models are very similar to GPT-2 in sentence coherence and completeness, which proves the effectiveness and reliability of our model.

3.3 Bad Case Analysis

Although most of reviews generated by our models are semantically correct and coherent, there are still some bad cases shown in 3. We classify the bad cases into 4 types: unusual word, repetition words, mistake in grammar & tense, and semantic contradiction.

- **Unusual word** takes place most likely in the reviews generated by character-level models, since character-level models can suffer from a higher word perplexity due to the nature of its char-to-char prediction.
- **Repetition words** always happen in word-level models. In the word-level model, a new word in a sequence is predicted based on the previous words. If previous word and target word are the same, then the model will keep predicting the same word without stopping.
- **Mistake in grammar & tense** is the most common problem, which appears in both character-level and word-level models with a very high frequency. That's because some users do not pay attention to the grammar and tense when typing the reviews so the dataset contains a lot of reviews with mistaken grammar and tense.
- **Semantic contradiction** is also a problem that usually happens, which is due to the limitation of the model that can not take a whole sentence into consideration.

Table 3: Example of bad cases generated by the models

Types	**Examples of bad cases**
Unusual word	Food in Kateron is unbelieviansa winery tasting room.
Repetition word	The environment is exasperating exasperating exasperating exasperating exasperating exasperating...
Mistake in grammar and tense	It was small, has large back room and a small video board if I live in.
Semantic contradiction	The sunflower seeds will perk the taste of the rest of the food, and it was so mediocre that I had to give a tip that it was frozen in the middle.

To conclude, our experimental results demonstrate that even two-layer RNNs are a fully feasible way of generating grammatically correct, convincing fake restaurant reviews. The performances of

our character-level and word-level models are strong enough to generate review sentence with coherence and completeness. Compared to character-level model, the word-level model can generate more complicated, coherent and meaningful sentences. Also, we proved that initializing the embedding layer of the word-level model with pre-trained Word2vec weights can significantly reduce the training time and slightly improve the performance, compared to the training using randomly generated weights.

4 Team Contributions

Tianxiao Hu was responsible for conducting the project roadmap, data preprocessing and vanilla two-layer LSTM model.
Yi-Fang Lee was responsible for the attention-based LSTM model training and problem background.
Zhenbang Chen was responsible for the word-level model and the pre-trained weights experiments.
Wenyuan Gao was responsible for the GPT-2 model training and experimental result analysis.
A percentage breakdown of contributions is listed as Table 4

Table 4: Team contributions by percentage

Contributions %			
Tianxiao Hu	Yi-Fang Lee	Zhenbang Chen	Wenyuan Gao
27.25%	21.75%	27%	24%

References

[1] Y. Yao, B. Viswanath, J. Cryan, H. Zheng, and B. Y. Zhao, "Automated crowdturfing attacks and defenses in online review systems," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1143–1158.

[2] N. J. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[3] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "What yelp fake review filter might be doing?" in *Seventh international AAAI conference on weblogs and social media*, 2013.

[4] Yelp.com. (2019) Yelp dataset. [Online]. Available: https://www.yelp.com/dataset

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[8] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015, pp. 2342–2350.

[9] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, p. 8, 2019.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.