Social Media Text Generation Based on Neural Network Model

Jiarun Cao
Digital Media Research Institute,
Beijing Institute of Technology
Beijing,China

18634988899,+86
Jiaruncao.china@Gmail.com

Chongwen Wang
Digital Media Research Institute,
Beijing Institute of Technology
Beijing,China

(010)68914044,+86
wcwzzw@bit.edu.cn

ABSTRACT

The social media text is increasing rapidly in recent years. With this background, natural language generation technology is mature enough for implementing an NLG system in some general field, but the circumstance is difficult in the social media field because of the linguistic arbitrariness. This paper presents a neural network model building a social media NLG system. Compared with state-of-art model, our system outperforms the existing NLG system in the social media field.

CCS Concepts

• Artificial intelligence→Natural language processing→Natural language generation• Human-centered computing→Human computer interaction(HCI).

Keywords

Natural Language Generation; Neural Network Model; Social Media application

1. INTRODUCTION

Recently, the idea of building a readable and realistic and NLG system about a social media text has come closer to reality principally through two methods[1]. The first method is the advent of deep learning techniques (Graves A et al., 2016), which utilizes multi layer RNN structure with LSTM unit. The second one is traditional method based on Hidden Markov Model (Vogel S, et al., 1996)[2].

However, these two kinds of method respectively takes some disadvantages. On the one hand, HMM method only depends on the state of each object and its corresponding observation, while the sequence problem not only associated with a single word, but also the length of the observed sequence and the context of the word.

On the other hand, recurrent neural networks model are known to have problems dealing with such long-range dependencies. Architectures like LSTMs should be able to deal with this, but in practice long-range dependencies are still problematic. Therefore, it is easy to lose useful information with a encoder-decoder structure in such a long sequence.

To solve the problem described above, This paper presents a three layer RNN structure with LSTM units and attention mechanism. With an attention mechanism we no longer try encode the full source sentence into a fixed-length vector. Specifically, attention mechanism gives all words a weight value , it helps the model learn what to attend to based on the input sentence and what it has produced so far.

To sum up, this paper makes the following contributions:

1) Introduce a three layer RNN model to effectively extract the linguistic feature.

2) Replace the basic RNN units with the LSTM units to alleviate the long-term dependency problem.

3) Introduce a encoder-encoder structure and add attention mechanism to the model. It helps to learn the meaningful information and throw away the useless words.

2. RELATED WORKS AND THEORIES

2.1 Recurrent Neural Network

A recurrent neural network (RNN) is a type of neural network architecture particularly suited for modeling sequential phenomena. At each time step t, an RNN takes the input vector $X_t = R^n$ and the hidden state vector $S_{t-1} \in R^m$ and produces the next hidden state S_t by applying the following recursive operation:

$$S_t = f\left(U_{X_t} + WS_{t-1} + b\right) \quad (1)$$

If here $U \in R^{m\times n}, W \in R^{m\times m}, b \in R^m$ are parameters of an affine transformation and fis an element-wise nonlinearity. In theory the RNN can summarize all historical information up to time t with the hidden state S_t. The chain characteristics reveal that RNN is essentially related to the sequence and the list. They are the most natural neural networks for this kind of data. However, learning long-range dependencies with a vanilla RNN is difficult due to vanish and exploding gradients[3].

2.2 Long Short-Term Memory

Long short-term memory (LSTM) addresses the problem of learning long range dependencies. LSTM model is no different from the general RNN structure, except with the hidden layers. All RNN have a repetitive chain form of neural network modules. In the standard RNN, this duplication has only a very simple structure, such as a tanh layer. The LSTM has the same structure, but the repeating module is a different structure. Instead of a single neural network layer, there are four layers interacting in a very specific way.

In the LSTM, first of all, we need to think about what information is discarded from the cell state, which is called “forget gate”. The door reads S_{t-1} and x_{t-1}, and outputs a value C_{t-1} to between 0 and 1 via the sigmoid function. one means “complete reservation", zero means "completely abandoned" :

$$f_{t-1} = g(W_f \times [h_{t-1}, x_t] + b_f) \quad (2)$$

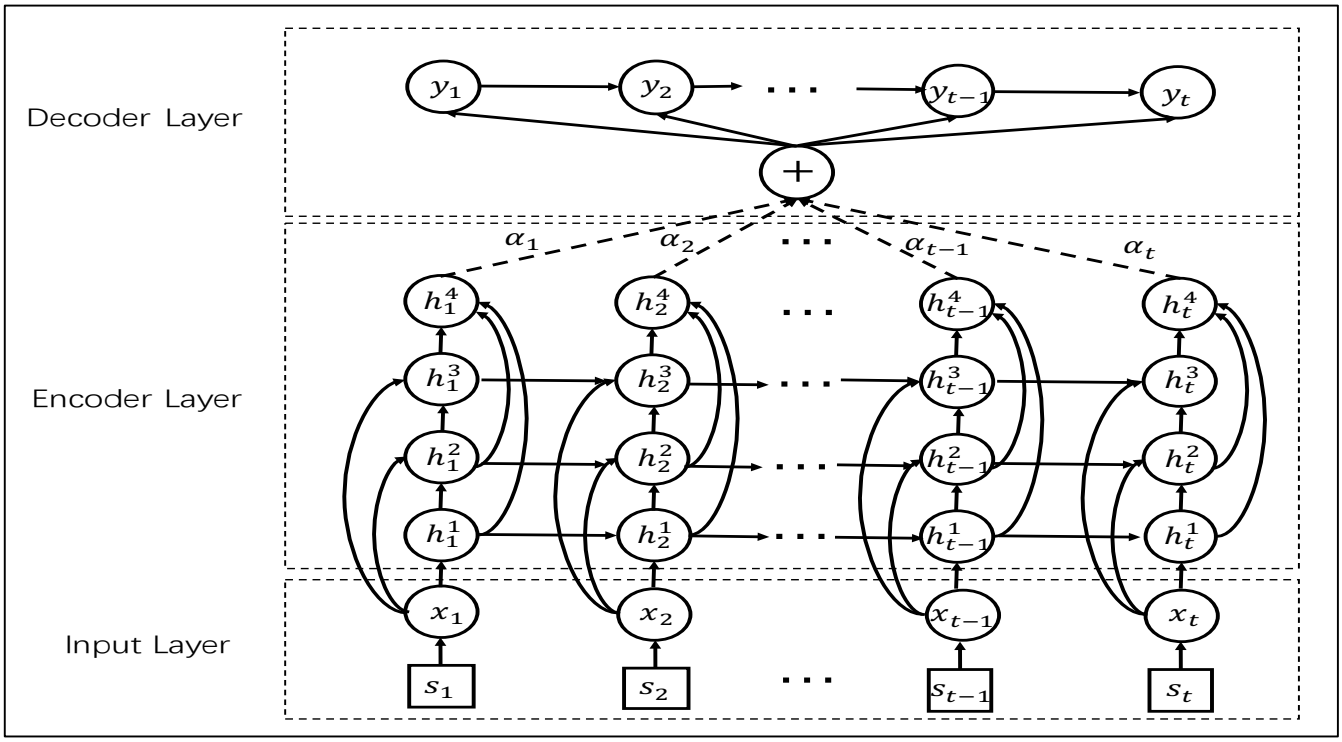

figure 1:Model Structure Overview

The next step is to determine what new information should be stored in the cell state. In which, "input gate" determines what value will be updated, and the tanh layer creates a new vector $\hat{C}_t$ into the cell state :

$$i_t = \text{sigmoid}(W_i \times [h_{t-1}, x_t] + b_i) \quad (3)$$

$$\hat{C}_t = \tanh(W_c \times [h_{t-1}, x_t] + b_c) \quad (4)$$

After that, update the cell state:

$$C_t = f_t \times C_{t-1} + i_t \times \hat{C}_t \quad (5)$$

Finally, we need to determine the value of the "output gate". This output is based on our cell state. We run a sigmoid layer to determine which part of the output of the cell state, then, treat cell state by tanh (get a value between 0 to 1) and put it multiplied with an sigmoid output, eventually we will only output the part which are determined :

$$o_t = \text{sigmoid}(W_o \times [h_{t-1}, x_t] + b_o) \quad (6)$$

$$h_t = o_t \times \tanh(C_t) \quad (7)$$

In the LSTM, first of all, we need to think about what information is discarded from the cell state, which is called

2.3 Word2Vec Embedding

It is usually necessary to mathematically transform the word to a vector.Mikolov introduced word2vec[4], a novel word embedding procedure. Their model learns a vector representation for each word using a (shallow) neural network language model. Specifically, they propose a neural network architecture (the skip-gram model) that consists of an input layer, a projection layer, and an output layer to predict nearby words. Each word vector is trained to maximize the log probability of neighboring words in corpus[5].

3. MODEL

The goal of a end2end language model is to sequentially predict the next words[6]. As shown in figure1, our model contains three parts of that:

3.1 Input Layer

The input layer is designed to convert raw words to respective semantic representations. As shown in figure 3, we are given a word sequence $\{s_1, s_2, \dots, s_{t-1}, s_t\}$, the input layer summarizes the word sequence with a vector with fixed length, which is determined by the number of word feature in the word2vec training processing. In this paper, we use pre-trained word vectors, GloVe[6],to obtain the word embedding $\{x_1, x_2, \dots, x_{t-1}, x_t\}$.

3.2 Encoder Layer

We use a Long Short-Term Memory Network(LSTM) which is presented by (Hochreiter&Schmidhuber,1997)consequently on top of the embedding provided by the previous layers to model the temporal interactions between words[7].

Specifically, the hidden layer activations are computed by iterating the following equations from $t = 1$ to t and from $n = 2$ to n :

$$i_t = W_{x_i} x_t + W_{h_i} h_{t_1} + W_{c_i} c_{t_1} + b_i \quad (8)$$

$$f_t = W_{x_f} x_t + W_{h_f} h_{t_1} + W_{c_f} c_{t_1} + b_f \quad (9)$$

$$c_t = f_t c_{t_1} + i_t \cdot \tanh\left(W_{x_c} x_t + W_{h_c} h_{t_1} + b_c\right) \quad (10)$$

$$o_t = W_{x_o} x_t + W_{h_o} h_{t_1} + W_{c_o} c_{t_1} + b_o \quad (11)$$

Therefore, the output of encoder layer is:

$$h_t = o_t \cdot \tanh(c_t) \quad (12)$$

1. 科比男篮，所有的球员，打球的时候就不知道怎么说，这就是纯粹的篮球，来自@腾讯新闻客户端
2017年12月14日，CBA联赛。北京队主场迎战新疆队。新浪体育 李欣 / 摄 @中国篮球 秒拍视频
Translation: Kobe Bryant basketball, all the players, don't know how to say, when I was playing this is purely a basketball, from @ Tencent news client on December 14, 2017, CBA league. Beijing hosts New Territories. Sina sports li xin/photo @china basketball second beat video.

2. 王源加油 我们一起看着你长大 加油！！！
#王源1108十六岁生日快乐##王源##王源银幕首演爵迹##遇见1
Translation: Come on, come on, let's watch you grow up!!!
#1108 Yuan Wang happy birthday of 16 # # # Yuan Wang # # screen debut jenn-air meet 1 # #

3. 浙江队赢得了比赛，最终比分108-121。恭喜！#科比最棒！
Translation: Zhe Jiang team wins the game finally, the ending score is 108-121.Congratulations!#Kobe is best!

figure 2:Sample of Generating Blog

3.3 Decoder Layer

To alleviate the long-term dependency and extract more useful words , we introduce attention mechanism, with an additional gate to determine the importance of information in the word sequence. Given encoder layer output h_t , the decoder layer generates sentence representation as follows:

$$v_t = \mathrm{LSTM}(v_{t-1}, c_t) \tag{13}$$

where $c_t = \mathrm{att}(v, v_t)$ is an attention pooling vector of the whole sequence:

$$s_i = v^t \cdot \tanh\left(W_v v_i + W'_v v_t\right) \tag{14}$$

$$a_i = \exp(s^t) / \sum_{i=1}^{t} \exp(s_i) \tag{15}$$

$$c_t = \sum_{i=1}^{t} a_i v_t \tag{16}$$

After that, we get the final result of generating word sequence:

$$y_t = \mathrm{softmax}(W_d v_t + b_t) \tag{17}$$

4. EXPERIMENT

4.1 Dataset

In this paper, our data source is collected from Sina Weibo. We used Python to write crawlers, crawling 60000 microblogs with keywords equal to “Kobe Bryant”(a famous NBA star), within 50,000 for training models and 10,000 for test results. Because of the micro blog’s timeliness, the crawling data under a certain keyword may be greatly different in different period, so we split training set and test set for one day time granularity, divided daily crawling data by 5:1.

At the same time, this paper uses the Chinese NLP library named “jieba”,and did some preprocessing data cleaning work: getting rid of some unusual stop words and incorrect tweets about the format (including too many Spaces and line breaks), finally the processed training set is 50000 sequence of words.

4.2 Training Parameter

Using TensorFlow to build the model mentioned above. All the parameters including the word embedding are initialized by randomly sampling from a uniform distribution $[-0.1, 0.1]$. The size of hidden layers units is 1000. No momentum and weight decay are used. We train the model with stochastic gradient descent and the gradients which has l_2 norm larger than 40 are clipped, learning rate being controlled by AdaDelta[8].

4.3 Decoder Layer

We control the length of the generated text to be about 50 to 80 words, which is similar to the average length of the microblogs. The first input character is null to ensure that the generated text is completely random. Sample from the model is shown as figure2.

5. RESULT AND ANALYSIS

5.1 Existing Method

There are two existing method to generate the blog words sequence:

The first one is traditional method based on HMM. Hidden Markov Model(HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved states. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states.

The second one is state-of-art method called Char-RNN model. Char-RNN model is present by (Graves A et al., 2013)[10],which contains a three hidden layer RNN model and uses the character-level representation of the sequence.

5.2 Evaluation Metrics

We compare our model with HMM model and Char-RNN model. To prove the effectiveness of our model, we use two ways to report the performance of the model: Perplexity and LSI similarity.

5.3 Perplexity

Perplexity is used in the field of natural language processing (NLP) to measure the quality of language models. It mainly estimates the probability of a sentence appearing according to each word, and use the length of the sentence to normalize.

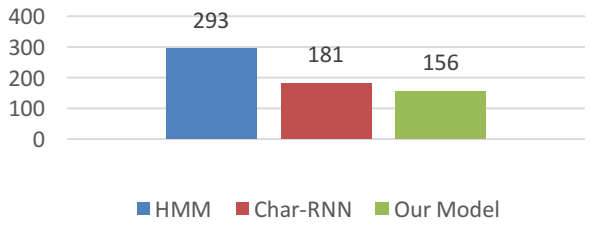

figure 3:Perplexity Comparison

Figure3 shows exact Perplexity score among three models. Neural network model largely outperforms the HMM model. And with the LSTM units and attention mechanism, our model is slightly better than state-of-art method(Char-RNN model) .

5.4 LSI Similarity

Latent Semantic Indexing(LSI)is a simple and practical thematic model. LSI is based on singular value decomposition (SVD) method to obtain the text topic. In SVD, we can get the relevancy of the document and the topic, the relevance of the word and the meaning of the word, and the relevance of the meaning and topic

In this paper, the LSI model is used to calculate the similarity between the generated text and the test text, then take it as the criterion for determining the result whether good or not.

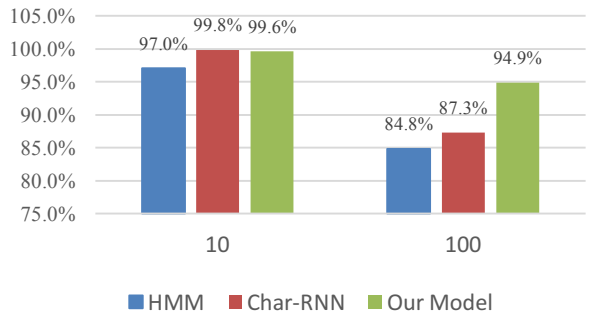

figure 4：Different Top Similarities numbers Comparison

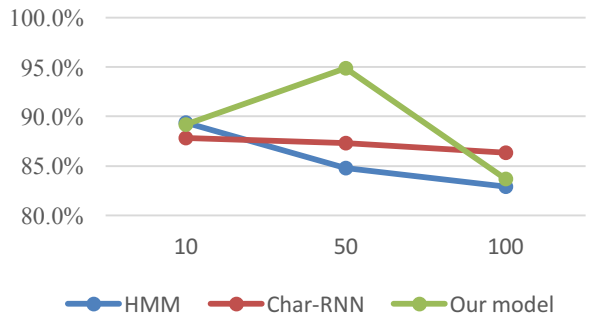

figure 5:Different Themes numbers Comparison

Figure4 shows the number of top similarities score among HMM, Char-RNN and our model, with a fixed themes number which is 50.If the number of top similarities is 10, neural network model is getting a higher score than HMM model, but our model is slightly lower than Char-RNN model. Since the number of top similarities is 100, our model is significantly better than other model.

With the top similarities number is 100, figure5 shows the result of different number of themes among three models. As the numbers of themes are 10 and 50, our model outperforms, while Char-RNN is getting a better result with the number of 100.

6. CONCLUSION

This paper discusses a kind of Weibo text generating method based on neural network model. First of all, we collect a large number of micro blog as raw data, and use the word embedding in order to map the word to a low dimensional vector space, and trained a RNN model with LSTM cells and introduce the attention mechanism. By comparing with the traditional Markov model and char-RNN model, it proves that the final result can improve performance.

But there are still plenty of things that can be improved, such as increasing the data set and cleaning the data more carefully. Moreover, model structure and adjusting parameters also have a efficient impact on the performance. These work will be done in the future research.

7. REFERENCES

[1] A. Alpher, , and J. P. N. Fotheringham-Smythe. Frobnication revisited. Journal of Foo, 13(1):234–778, 2003.

[2] Vogel S, Ney H, Tillmann C. HMM-based word alignment in statistical translation[C]// Conference on Computational Linguistics. Association for Computational Linguistics Morristown, 1996:836-841.

[3] A. Alpher. Frobnication. Journal of Foo, 12(1):234–778, 2002.).

[4] Authors. The frobnicatable foo filter, 2018. Face and Gesture 2018 submission ID 324. Supplied as additional material efg324.pdf.

[5] Authors. Frobnication tutorial, 2018. Supplied as additional material tr.pdf.

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In EMNLP, 2014.

[7] Gers F A, Schraudolph N N. Learning precise timing with lstm recurrent networks[M]. JMLR.org, 2003.

[8] Zeiler M D. ADADELTA: An Adaptive Learning Rate Method[J]. Computer Science, 2012.

[9] Sutskever I, Martens J, Hinton G E. Generating Text with Recurrent Neural Networks[C]// International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July. DBLP, 2011:1017-1024.

[10] Graves A. Generating Sequences With Recurrent Neural Networks[J]. Computer Science, 2013.

Columns on Last Page Should Be Made As Close As Possible to Equal Length

- **The below form will not be published, but it will help us to understand your paper better**

Authors' background

Name	Email	Position (Prof , Assoc. Prof. etc.)	Research Field	Homepage URL
JiarunCao	**Jiaruncao.china@gmail.com**	**Student**	**NLP**	**www.caojiarun.com**
Chongwen Wang	**wcwzzw@bit.edu.cn**	**Prof.**	**Digital Media**	**http://ss.bit.edu.cn/szdw/szdwlb/81121.htm**