The Prediction of Character Based on Recurrent Neural Network Language Model

Zejian Shi
School of Computer Science,
Communication University of China
Beijing, P.R. China
541838623@qq.com

Minyong Shi
School of Computer Science,
Communication University of China
Beijing, P.R. China
myshi@cuc.edu.cn

Chunfang Li
School of Computer Science,
Communication University of China
Beijing, P.R. China

***Abstract*—This paper mainly talks about the Recurrent Neural Network and introduces a more effective neural network model named LSTM. Then, the paper recommends a special language model based on Recurrent Neural Network. With the help of LSTM and RNN language models, program can predict the next character after a certain character. The main purpose of this paper is to compare the LSTM model with the standard RNN model and see their results in character prediction. So we can see the huge potential of Recurrent Neural Network Language Model in the field of character prediction.**

Keywords—RNN; LSTM; language model; character prediction

I. Introduction

Before introducing the Recurrent Neural Network, you should know what is the neural network and how does it develop. The human brain has a complicated structure at birth and possesses an ability which we usually called the "experience" to set up its own rules. So an evolving neural system is synonymous with plastic brain. If regarding the neural network as a kind of self-adaption machine, there can be a definition as follows that the neural network is a large-scale parallel distributed processor consisted of simple processing units. The neural network has features of storing experience and using experience. And the neural network is similar to the brain in two aspects: getting knowledge from external environment through learning process and storing knowledge through connection strength among nerve cells.

Neural network technology originated from perceptron in the 1950s. Perceptron is a dichotomy linear model which has a input layer, a output layer and a hidden layer. Cause monolayer is a linear model, it has a very serious problem that it can't match "XOR" operation. Therefore, single-layer perceptron doesn't has too much value.

In the 1980s, Rumelhart, Williams, Hinton et al created multilayer perceptron and solved the shortcoming that single-layer perceptron couldn't simulate "XOR". Multilayer perceptron uses "sigmoid" or "tanh" function to simulate the respond of nerve cells to stimulation and uses BP algorithm as its training algorithm. This is feed-forward neural network that we usually say. The layer number of neural network decides the ability of depicting reality directly, which is reflected in the Bengio's paper: if a function matched by a k-layer network simply is matched by k-1 layer, it may need exponent level compute nodes [1]. But with the neural network layers becoming deeper, optimization function is more and more prone to local optimal solution and gradient disappear phenomenon is more serious.

Until 2006, Hinton used the method of pre-training to relieve the local optimal solution problem and pushed hidden layers to 7-layer [2], which made the deep neural networks acquire ground-breaking development. In 2015. the emergence of highway network [3] and deep neural residual [4] solved the problem of gradient disappearing. But the increase of layers brought a new question that the number of arguments is too large.

Then we need an another neural network model to solve the problem that multilayer neural networks have too many arguments. It is Convolutional Neural Network which called CNN for short. CNN is a kind of feedforward neural network. And its artificial neuron can response part of around units in covered range. Convolutional Neural Network limits the number of parameters and is able to explore the characteristics of the local structure. But the mentioned neural network above can't simulate time series models, such as natural language processing, speech recognition, handwriting recognition and so on. In order to meet this demand, the Recurrent Neural Network appeared.

Like most of the neural networks, the Recurrent Neural Network is not something new. Its nodes directionally connect with each other into a loop. This internal state can show the dynamic timing behavior of the network. Differing from feedforward neural networks, RNN uses its internal memory to handle any input timing sequence, which make it easier to handle handwriting recognition and speech recognition. In the early 1990s, gradient disappearing is the biggest problem for Recurrent Neural Network. In the mid-1990s, the German scholar Sepp Hochreiter and Juergen Schmidhuber raised a special kind of Recurrent Neural Network named LSTM to solve gradient disappearing problem [5].

The full name of LSTM is Long Short Term Memory, which can selectively store and discard the information in the hidden layer of neural networks. In other word, it can decide the information storage time in neuron. The emergence of LSTM figured out the gradient appearing problem well and promoted the development of Recurrent Neural Network.

II. Theory of RNN

A. Explanation of Standaard RNN

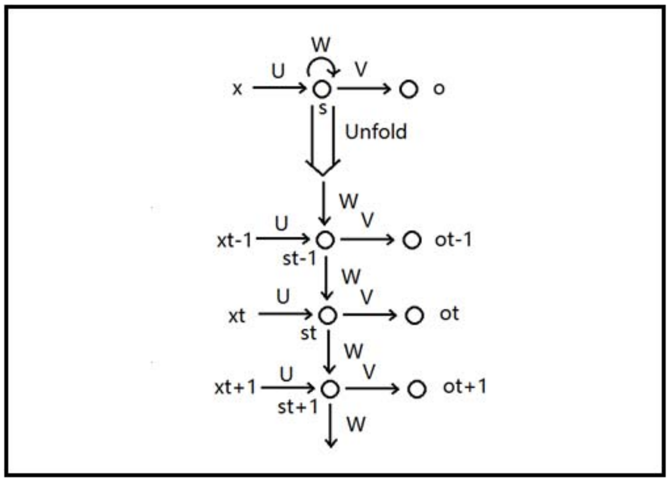


Fig. 1. The expansion graph of RNN

As wo can see in the Fig. 1, it is the process that a standard recurrent neural network unfolds into a network. It has a input layer, a output layer and a hidden layer. The "x" means the input layer. The "s" means the hidden layer. And the "o" means the output layer. It is obvious that the hidden layer is calculated by the current input layer and previous hidden layer. So we can understand that the key of this graph is the cyclicity of RNN. The formula showed in the graph is as follows. In this formula, the function "f" is nonlinear such as "tanh" function.

$$s_t = f(U \cdot x_t + W \cdot s_{t-1}) \tag{1}$$

$$o_t = g(V \cdot s_t) \tag{2}$$

B. Explanation of LSTM

LSTM is a special recurrent neural network. Because the repeating module of ordinary RNN only has a simple structure and LSTM uses a more complex structure to replace it. Then let's compare the structure of LSTM with the structure of standard RNN. The explanation of LSTM combines my own understanding with an article named "Understanding LSTM Networks" [6].

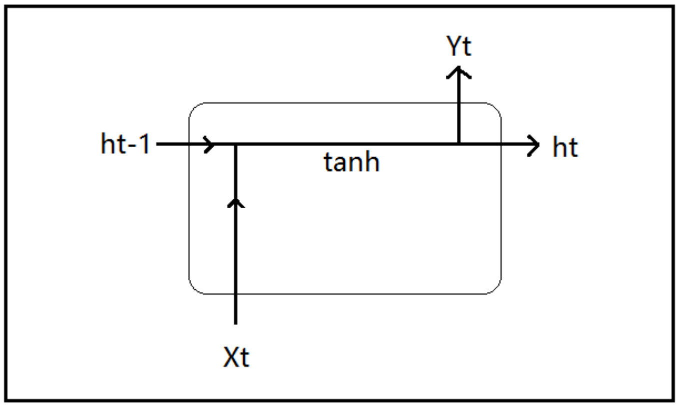


Fig. 2. The construction of standard RNN

This is the repeating structure inside the standard RNN. It is very easy to understand. Then I will introduce the structure inside the LSTM from four parts.

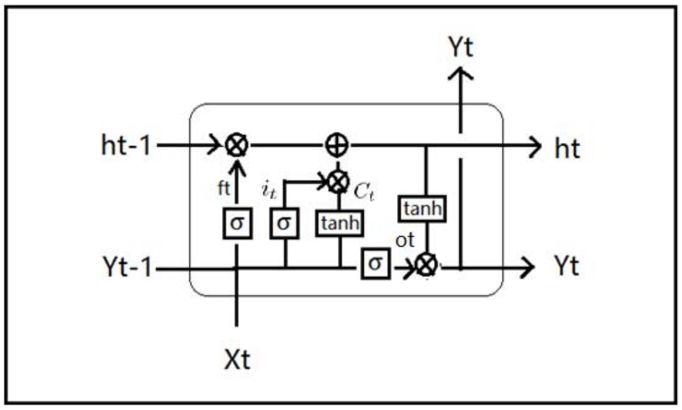


Fig. 3. The construction of LSTM

$$f_t = \sigma\left(W_f \cdot [Y_{t-1}, x_t] + b_f\right) \tag{3}$$

This part is to decide what information we are going to throw away from the cell state. The key to LSTM is the cell state, which we can regard it as hidden layer state. It runs straight down the entire chain with only some minor linear interactions. Therefore information can be easily maintained during transmission. Sometime the information in the cell state is unnecessary. And the LSTM will remove the information in the cell state through "forget gate layer". The gate is a selective way of controlling information transmission which is "sigmoid" function It outputs a number between 0 and 1. The "1" means "completely keep this" and the "0" means "completely get rid of this". It is very helpful for our language model trying to predict the next word based on all the previous ones. Because some information from the previous ones may be of no worth.

$$i_t = \sigma(W_i \cdot [Y_{t-1}, x_t] + b_i) \tag{4}$$

$$C_t = tanh(W_C \cdot [Y_{t-1}, x_t] + b_C) \tag{5}$$

This part is to decide what new information we are going to store in the cell state. Sometimes we want to add some new information to the cell state in our language model. The "sigmoid" layer decides which values to update and the "tanh" layer creates a vector of new candidate values that can be added to the state.

$$h_t = f_t \cdot h_{t-1} + i_t \cdot C_t \tag{6}$$

This part is to update the old cell state combining what is to delete with what is to add. In other word, we drop useless information and add new information in our language model.

$$o_t = \sigma(W_o \cdot [Y_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t \cdot \tanh(h_t) \tag{8}$$

Final part is to decide what to output. It is the time that we output the information we need. In our language model, since that we want to know what is the next word, it may output the characteristic of the next word.

III. Construction of Language Models

We already have some knowledge of RNN. Then we want to construct the language model for RNN in order to generate new text. Specifically, RNN will be inputted a lot of text data and construct a model that predict the next character of a known character. Generally speaking, the function of language models is to judge if a word is said by people. It is a most precise task that constructing a language model from natural text. Because, the program need to process large data, do statistics and analysis. In other words, this job is to get the vector of output words without supervision from a mass of unmarked text.

A. Vector Representations of Words

If a word wants to be processed by program, the first step is to find a way of transferring words into math representation. This is the vector representations of words. Since we want to train RNN character-level language models, the One-hot Representation is the most common way for our project. This approach encodes each character into a vector using 1-of-k encoding which means there are all zero except for a single one at the index of the character in the vocabulary. For example, the character "a" is encoded as [1 0 0 0 ...]. This way is very concise.

But if we want to train RNN word-level language models, the dimension of vector will be too large. In deep learning, the Distributed Representation is the most useful way to our language models. The word is encoded as a low dimensional vector in this way which is usually called Word Embedding. What's more, it shortens the distance among similar words. The distance can be measured by the traditional Euclidean Distance or the cosine of the angle. When constructing a high dimensional language model, tradition word representation will cause dimension disaster and unacceptable complexity. It is why the Distributed Representation is popular in deep learning. Mikolov introduced the Distributed Representations of Words in his paper and said that learning good vector representations for millions of phrases was possible with Word Embedding [7].

B. Training of Language Models

Some people have tried to use artificial neural networks in statistical language models. For instance, Bengio used a three-layer network to construct language model in 2001 [8]. As we can see in Fig. 4, w_{t-n+1},...,w_{t-2},w_{t-1}represent known words whose number is n-1. Now we need to predict the next word w_t based on known words. C(w) means the vector of words which is a V*m matrix. V means the total number of words and m means the dimension of words vector. So the processing of w to C(w) means taking up a line in matrix C. The first layer is the input layer. It combines these n-1 vectors into a (n-1)*m dimension vector. The second layer is the hidden layer. It is computed as standard neural networks. The third layer is the output layer. The number of its nodes is V. Yi means the log probability that the next word is it. Finally, using softmax function transfers the Y into normalized probability.

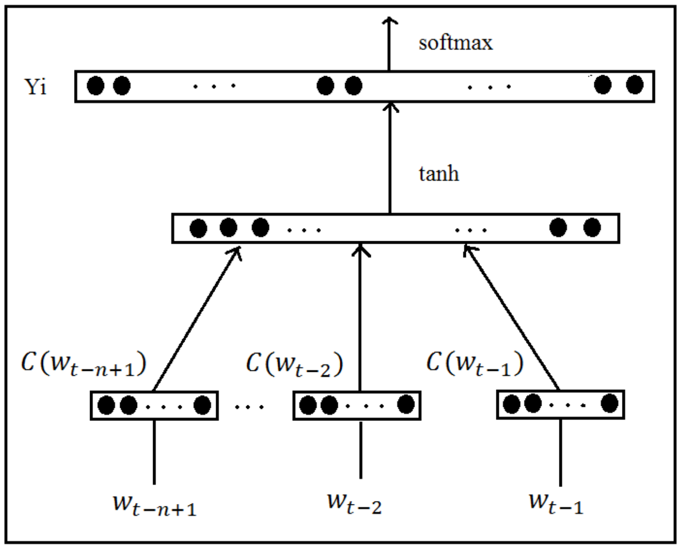


Fig. 4. The process of constructing neural network language models

Later, Tomas Mikolov used a new recurrent neural network based on language model with applications in speech recognition [9]. Although Recurrent Neural Network is different from the above neural network in structure, their theories are similar. The biggest advantage of Recurrent Neural Network is that it can make full use of all the above information to predict the next word.

IV. Amendment of Parameters

Now we have known the language model. But there are still two crucial tasks to do in order to optimize the parameters of our Recurrent Neural Network Language Models. The first one is loss function. The second one is backpropagation.

A. Loss Function

Assuming that we have got the target word according to initial parameters. Then we want to know the error about real word. So the loss function is necessary. Here we would like to minimize the average negative log probability of the target words. The function is as formula (9) which is not very difficult to implement.

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} ln p_{target_i} \tag{9}$$

B. Truncated Backpropagation

Since we have known the loss function, the next step is to optimize parameters and reduce the loss. Next we illustrate the principle of it through a simple example.

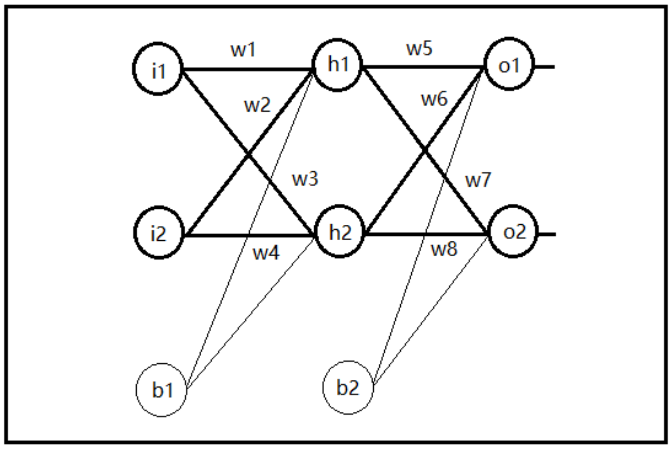


Fig. 5. The graph of artificial neural networks

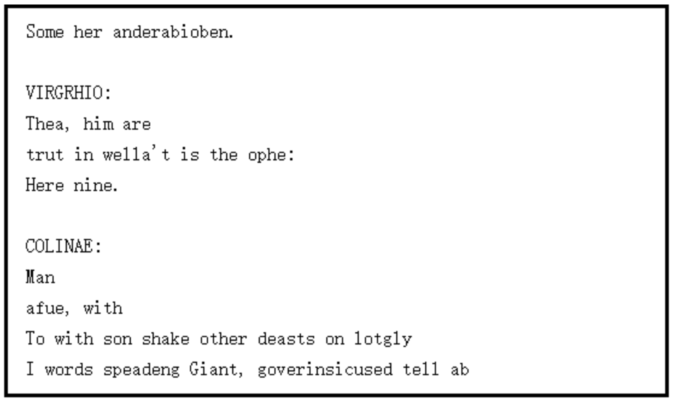
As we can see in the Fig. 5, i1 and i2 mean the input layer, h1 and h2 mean the hidden layer, o1 and o2 mean the output layer, b1 and b2 mean the intercept. Assuming that the total error is variance, now the job is to decrease the error. Then, we talk about how to update the weight. The formula is as follows. The character η is learning rate. Through many times of iteration, the E_{total} will be lower and lower.

$$E_{total} = \sum \frac{1}{2}(target - output)^2 \tag{10}$$

$$w_i' = w_i - \eta * \frac{\partial E_{total}}{\partial w_i} \tag{11}$$

V. Result And Analysis

I use two language models, language model based on standard RNN and language model based on LSTM, to process the same input text from a famous play. Fig. 6 shows the result of standard RNN after dozens of minutes. At the beginning of training, the output text looks like messy characters. But after training, the output text looks like words. Fig. 7 shows the result of LSTM. It uses three layers and 256 hidden nodes. We can see that the accuracy of words is improved greatly and it already has the structure of sentence and play.

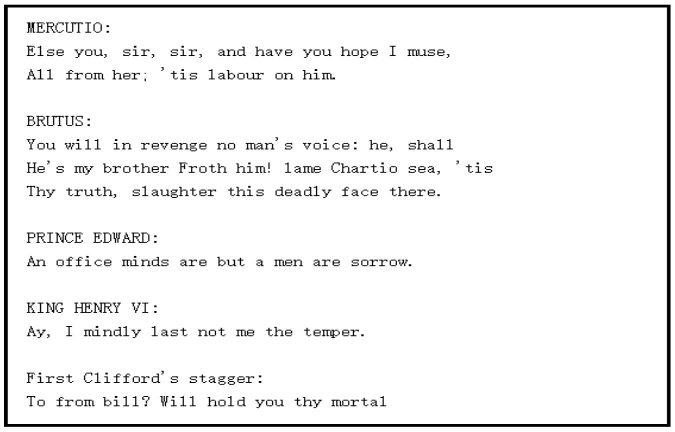


```
Some her anderabioben.

VIRGRHIO:
Thea, him are
trut in wella't is the ophe:
Here nine.

COLINAE:
Man
afue, with
To with son shake other deasts on lotgly
I words speadeng Giant, goverinsicused tell ab
```

Fig. 6. Result of standard RNN



```
MERCUTIO:
Else you, sir, sir, and have you hope I muse,
All from her; 'tis labour on him.

BRUTUS:
You will in revenge no man's voice: he, shall
He's my brother Froth him! lame Chartio sea, 'tis
Thy truth, slaughter this deadly face there.

PRINCE EDWARD:
An office minds are but a men are sorrow.

KING HENRY VI:
Ay, I mindly last not me the temper.

First Clifford's stagger:
To from bill? Will hold you thy mortal
```

Fig. 7. Result of LSTM

VI. Conclution And Future Work

This paper mainly tells the development of neural networks, the distinction between standard RNN and LSTM, and language models based on neural networks. The program is run by tensorflow platform, which shows the ability of RNN in sequential processing. For the later work, if the program uses RNN to implement a word-level language model, the result may be greatly improved.

Acknowledgment

This paper is partly supported by “the Excellent Young Teachers Training Project (the second level, Project number: YXJS201508)”, “Key Cultivation Engineering Project of Communication University of China (Project number: 3132016XNG1606 and 3132016XNG1608)”, “Cultural technological innovation project of Ministry of Culture of P. R. China (Project number: 2014-12)”, and partly supported by “The comprehensive reform project of computer science and technology, department of science and Engineering”. The research work was also supported by “Chaoyang District Science and Technology Project (CYXC1504)”.

References

[1] Y. Bengio, “Learning Deep Architectures for AI,” Foundations &Trends in Machine Learning, 2009, pp. 1-127.

[2] G.E. Hinton, R.R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” Science, 2006, pp. 504-507.

[3] K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition,” arXiv:1512.03385, 2015.

[4] R.K. Srivastava, K. Greff, J. Schmidhuber, “Highway networks,” arXiv:1505.00387, 2015.

[5] S. Hochreiter, J. Schmidhuber, “Long short-term memory,” Neural Computation, 1997, pp. 1735-1780.

[6] “Understanding LSTM Networks”. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, August 2015.

[7] T. Mikolov, I. Sutskever, K. Chen, et al, “Distributed Representations of Words and Phrases and their Compositionality,” Advances in Neural Information Processing Systems, 2013, pp. 3111-3119.

[8] Y. Bengio, H. Schwenk, J. Senécal, et al, “Neural Probabilistic Language Models,” Journal of Machine Learning Research, 2006, pp. 1137-1155.

[9] T. Mikolov, M. Karafiát, L. Burget, et al, “Recurrent neural network based language model,” IConference of the International Speech Communication Association, Makuhari, Chiba, Japan, DBLP, September.2010, pp. 1045-1048.