Prediction Popularity of Video Contents with Deep Recurrent Neural Network

Kyi Thar, Nguyen H. Tran, and Choong Seon Hong
Department of Computer Science and Engineering, Kyung Hee University.
Email: {kyithar, nguyenth, cshong}@khu.ac.kr

Abstract

In the next generation network architecture, in order to reduce the network's traffic and service delay, node (router, base station, access point, etc…) temporary stores the contents on its cache storage space and provides those stored contents to users, instead of downloading contents from original servers. Thus, cache decision (the decision to store and remove contents from the cache storage space) becomes an important task to improve the network's performance. Cache decision is easy when nodes know the popularity of the contents. In reality, it is not easy to know the actual popularity of the contents. So, the content's popularity prediction becomes the most challenging task to make cache decision efficiently. Therefore, in this paper, we proposed the deep recurrent neural network based video content's popularity prediction scheme to support cache decision process. Finally, we have tested the proposed scheme by using Tensorflow which is the open source software library for machine intelligence.

1. Introduction

According to the Cisco Visual Networking Index (VIN), the traffic of watching video through the Internet is increasing exponentially [1]. Therefore, several architectures for next generation network (e.g. [2]) are proposed to reduce these increasing traffic and improve the performance of the network by adding the cache storage space at the nodes (routers, access point, base station and so on). Thus, the nodes can store the contents temporarily to satisfy requests for near future instead of retrieving again from the original servers. So, cache decision (which contents should store and remove from the cache space) becomes important to improve the network performance. To make a cache decision efficiently, the nodes needs to know the popularity of the contents. But in reality, it is quite difficult to know the popularity of the contents. So, the content popularity perdition becomes the challenging task for the researchers.

There are several methods to predict the popularity of the contents [3][4][5]. In this paper, we apply the deep recurrent neural network to predict the content's popularity. The recurrent neural network (RNN) received incredible success in speech recognition, language modeling, translation, image captioning and collaborative filtering. Among the variants of RNN such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) are good for sequence prediction. Thus, it is also good for predicting the video content's popularity, where the incoming requests for these contents are in a sequential manner. So, in this paper, we apply LSTM [6] to predict the popularity of video contents. For the training and testing of our proposed scheme, we choose MovieLens dataset with 1 million requests. The input features to feed the system are the sequences of movie's information such as genres, the release date of movies. The output values from the prediction system are the predicted popularity scores of contents in the range from 0 to 1. The nodes can make cache decision by using these predicted scores, where the cache decision process is out of scope of this paper.

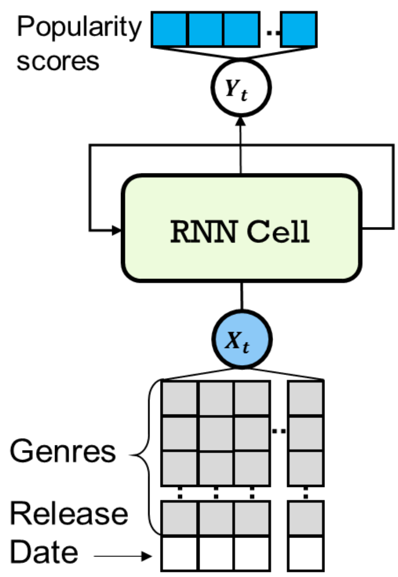

Figure 1 An unrolled recurrent neural network for predicting the video content popularity

2. System Model

An unrolled version of the RNN model uses in this paper is shown in Fig.1, where RNN cell is the memory cell which composed of several gates. The inputs to the system are the sequence of requests $X = \{x_1, \dots, x_T\}$, where x_t is the movie's related features such as genres, released date, etc... The output from the prediction system are $Y = \{y_1, \dots, y_T\}$, where y_t is the popularity score of the content. In this paper, we use LSTM [6] as a memory cell because it is designed to avoid the vanishing gradient problem and long-term dependency problem.

2.1. Long short-term memory (LSTM)

Each LSTM cell is managed by the several gates. While the input gate and the forget gate controls the input and the output of the system respectively. Especially, the forget gate controls the states of the cell. The forget gate is defined as,

$$\boldsymbol{f} = \boldsymbol{\sigma}\,(\boldsymbol{X}.\boldsymbol{W_f} + \boldsymbol{b_f}), \tag{1}$$

where $\boldsymbol{\sigma}$ is the sigmoid activation function, $\boldsymbol{X}$ is the input features, $\boldsymbol{W_f}$ is the weight for the forget gate and $\boldsymbol{b_f}$ is the bias for the forget gate. The update gate is denoted as,

$$\boldsymbol{u} = \boldsymbol{\sigma}\,(\boldsymbol{X}.\boldsymbol{W_u} + \boldsymbol{b_u}), \tag{2}$$

where $\boldsymbol{W_u}$ is the weight for the update gate and $\boldsymbol{b_u}$ is the bias for the forget gate. The result gate is define as,

$$\boldsymbol{r} = \boldsymbol{\sigma}\,(\boldsymbol{X}.\boldsymbol{W_r} + \boldsymbol{b_r}). \tag{3}$$

where $\boldsymbol{W_r}$ is the weight for the result gate and $\boldsymbol{b_r}$ is the bias for the result gate. The input gate is define as,

$$\boldsymbol{X'} = \boldsymbol{tanh}\,(\boldsymbol{X}.\boldsymbol{W_c} + \boldsymbol{b_c}), \tag{4}$$

where $\boldsymbol{W_c}$ is the weight for the result gate and $\boldsymbol{c}$ is the bias for the result gate. The memory from current block is denoted as,

$$\boldsymbol{C_t} = \boldsymbol{f}.\boldsymbol{C_{t-1}} + \boldsymbol{u}.\boldsymbol{X'}, \tag{5}$$

where $\boldsymbol{C_{t-1}}$ is the memory from the previous block. The output of the previous block is define as,

$$\boldsymbol{H_t} = \boldsymbol{r} * \boldsymbol{tanh}(\boldsymbol{C_t}). \tag{6}$$

The predicted output of the current block is denoted as

$$\boldsymbol{Y_t} = \boldsymbol{r} * \boldsymbol{softmax}(\boldsymbol{C_t}). \tag{7}$$

2.2. Performance matrices

In this section, we define the performance matrices to measure the performance of the proposed scheme. First, we measure the Mean Squared Error (MSE) as follows,

$$\boldsymbol{MSE} = \frac{\boldsymbol{1}}{\boldsymbol{n}} \sum_{i=1}^{n} (\boldsymbol{\hat{Y}_i} - \boldsymbol{Y_i})^{\boldsymbol{2}}, \tag{8}$$

where $\boldsymbol{\hat{Y}}$ is the predicted popularity score of the n predictions. Our goal is to minimize the error between predicted popularity scores and real popularity scores by updating the weight of the LSTM. Then, we define the accuracy (ACC) to measure how much accurate the proposed system by using following equation,

$$\boldsymbol{ACC} = \frac{\boldsymbol{1}}{\boldsymbol{n}} \sum_{i=1}^{n} (||\boldsymbol{\hat{Y}_i} - \boldsymbol{Y_i}|| < \boldsymbol{\gamma}), \tag{9}$$

where γ is the threshold value to count zero or one.

2.3. Minimize error

We use (Stochastic Gradient Descent) SGD with Backpropagation Through Time (BPTT) to minimize the total training error (MSE).

3. Performance evaluations

In this section, we present the properties of the data set, preprocessing processes. Then, we present results from testing. We run and test our proposed scheme by using Tensorflow version 1.1 [7], which is the open source software library for machine intelligence, on the window 10 environments. In order to run tensor flow on window environment, first, we install Anaconda which is also the open data science platform. Then, we create a virtual environment for Tensorflow and install Tensorflow on it. To preprocess the data set, we use pandas which is also the open source data analysis tools.

For the training and testing, we choose Movielens 1M [8] dataset, which includes 6040 users and 1,000,209 ratings over 3706 movies. Also, this dataset includes others information related to the users such as age, sex, occupation. Also, the data set includes movies related information such as released date and genre.

Then, we preprocess the input data. We feed the genre information, release date information of each movie and rating of the movie to the system.

Therefore, we turn genre information into one hot vector where genre information has 18 categories. We feed normalized values of the movie's released date and rating value to the system.

3.1 Results and discussion

Fig.3 shows the comparison of the training and testing error. The results show that the proposed scheme reduces the lost obviously from the step 60.

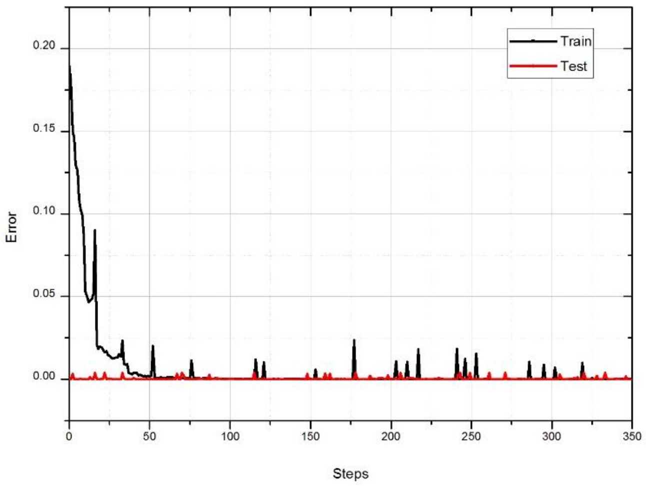

Figure 2 Comparison of train and test error

Fig.4 shows the comparison of accuracy between the training period and testing period. The results show that the accuracy reach around 90 percent at around 40 steps.

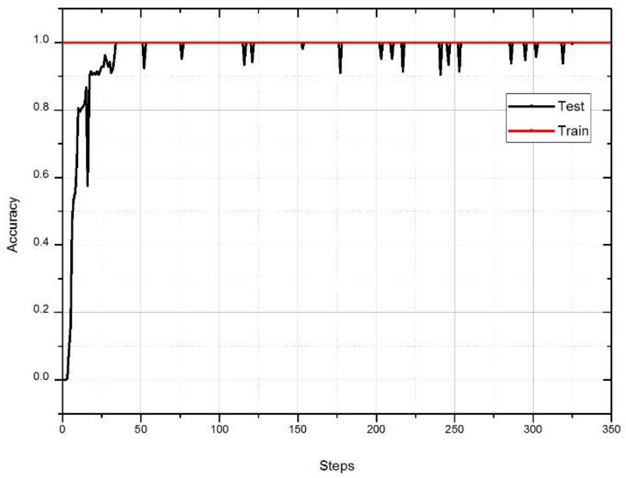

Figure 3 Comparison of train and testing accuracy

Parameters	Values
Input	20
Output	1
Number of hidden layers	100
Learning rate	0.2
Batch Size	100
DataSet	Movielens 1M
Training	60%
Validation	20%
Testing	20%

5. Conclusion

In this paper, we discussed the proposed scheme which predict the popularity of the video content by applying LSTM. We tested the proposed scheme by using Tensorflow with the MovieLens data set. As for the future work, we will apply the reinforcement learning with the recurrent neural network for the video contents predicting.

6. Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2013-0-00409, Research and Development of 5G Mobile Communications Technologies using CCN-based Multi-dimensional Scalability) *Dr. CS Hong is the corresponding author.

7. References

[1] http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 1-12. ACM, 2009.

[3] Ouyang, Shuxin, Chenyu Li, and Xueming Li. "A Peek Into the Future: Predicting the Popularity of Online Videos." IEEE Access 4 (2016): 3026-3033.

[4] Li, Chenyu, Jun Liu, and Shuxin Ouyang. "Characterizing and predicting the popularity of online videos." IEEE Access 4 (2016): 1630-1641.

[5] Devooght, Robin, and Hugues Bersini. "Collaborative filtering with recurrent neural networks." arXiv preprint arXiv:1608.07400, 2016.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," IEEE Transactions on Neural Networks, vol. 9, no. 8, pp.1735–1780, Nov. 1997.

[7] https://www.tensorflow.org/

[8] https://grouplens.org/datasets/movielens/