A Data-driven Approach for Performance Evaluation for Cache Server Groups in Content Delivery Network

None

Radarweg 29, Amsterdam

Elsevier Inc[a,b], Global Customer Service[b,]*

[a] *1600 John F Kennedy Boulevard, Philadelphia*
[b] *360 Park Avenue South, New York*

Abstract

In industry, Contnet Delivery Network(CDN) Service providers are increasingly using data-driven mechanisms to build their performance model of their service-providing systems. To build a model to accurately describe the performance of the existing infrastructure is very crucial to make resource management decisions. Conventional approaches that use hand-tuned parameters or linear models have their drawback. Recently, data-driven paradigm have been shown to greatly outperform traditional methods in modling complex systems. We design a approach that using these techniques to build a performance model for CDN cache server groups. We use deep LSTM autoencoder to caputrue the temporal structures from the high-dimensional monitoring data, and use a deep neural network to predict the reach rate, which is a client QoS measuement from the CDN service providers perspective. The experimental results have shown that our model is able to outperform state-of-the-arts models.

Keywords: *edge computing, deep learning, content delivery networks, sequence learning, predictive analysis, high dimensional data, prognosis*

1. Introduction

There is a trend [1] [2] that using data-driven methods to model complex networked systems. Traditional approach typically use simple huristics. These methods have several drawbacks.They cannot accurately reflect and oversimplified the complex systems due to the lack of knowledge of real-word environment. Driven by the opportunity to collect and analyze data (e.g., application quality measurement from end users), many recent proposals have demonstrated the promise of using deep learning to characterize and optimize networked systems. Drawing parralel from the success of deep-learning on pattern recognizaition, instead of using empirical non-linear learning model to descirbe the complex interaction of different features, we use deep learning methods and treat networked systems as a black-box.

A content delivery network (CDN) is a globally distributed network system deployed across the Internet. Composed with geographically distributed cache servers, CDNs deliver cached content to customers worldwide based on their geographic locations. Extensively using cache servers, content delivery over CDN has low latency and high reliability, and supports better quality of experience.

The CDN Service providers are increasingly using data-driven mechanisms to build their performance model of their service-providing systems. To build a model to accurately provice an understanding of the performance of the existing infrastructure such as the health of cache groups and network status, is very crucial to make resource management decisions including content placement, network traffic scheduling, load banlance of the CDN network. Modeling all available physical resources, we can maximize a resource utilization in terms of service quality, cost, profit, etc.

Generally CDN provicers don't have direct measurement from the clients (the logs from video players, web browser that can show the QoE of clients). Instead, they use the indirect measurement reach ratedefined in Section II which is collected and analyzed from the log of the HA proxy of CDN cache groups, which is . The computation of reach rateis done offline . In order to enable themselves make resource manangement decisions in real time, the CDN providers have to use the metrics that can be collected in the real time to infer the reach rate.

Cache server group can be characterised as multi-dimensional, highly non-linear, time variant vectors. The metrics that collected from members of the CDN cache server groups are sequence data that are measured every minute, which

*Corresponding author
Email address: support@elsevier.com (Global Customer Service)
URL: www.elsevier.com (Elsevier Inc)

have hundreds of dimensions. The state-of-art methods are typically using simple huristics, which are over-simplifed and baised due to the human experience.

We frame our problem as a sequence learning problem, which consists of stages: (1) feature engineering (2) representaion learning by lstm auto-encoder (3) fully connected network/ svm/ other black-box machine learning algorithm to output the predictions. lstm,lstm auto encoder and decoder

Our main contributions are listed below:

- data-driven approach
- performance modeling as sequence modeling problem
- anomaly detection(Collective Anomalies a) and prediction

The remain organization of this paper is as follows. In Section II, we first introduce the formulation of performance evaluation problem and then introduce our LSTM based structure. In Section III, we introduce reach-rate prediction algorithms based on the auto-encoder and decoder. In Section IV, we demonstrate performance improvements over baseline models. Finally, we provide concluding remarks in Section V.

2. Background

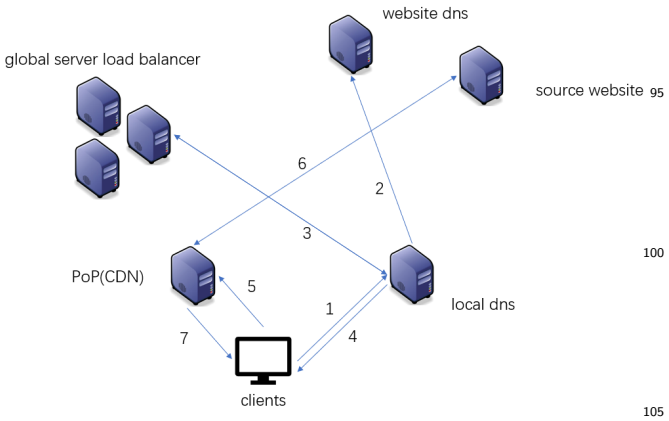

Figure 1: The basic working procedure of CDN

A content delivery network or content distribution network (CDN) (figure 1) is a geographically distributed network of cache servers. CDN helps content provider deliver web pages and other multi-media content to the clients, based on the locations of the clients and cache servers nearby the clients. The basic working procedure is as follows. Step 1client send a request to local dns. Step 2: the DNS finds the CNAME and redirects the request to the gslb (global sever load balance). Step 3: the local DNS server sends request to the glsb and glsb returns the ip address of CDN servers based on the scheduling policy. Step 4: the local DNS return the ip address to the clients. Step 5: fetch

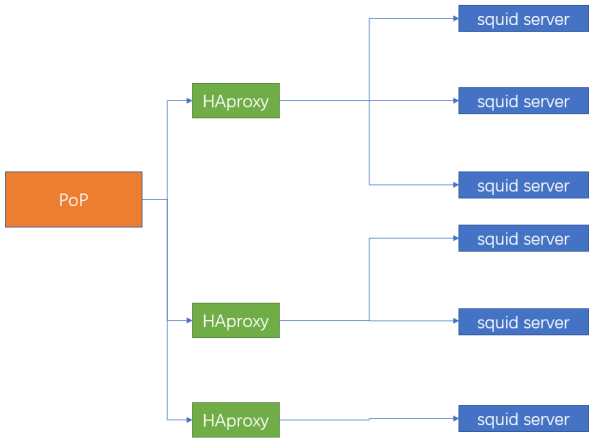

Figure 2: The figure

A CDN cache server group is the basic resource scheduling unit for CDN. A CDN cache group is a load banlanced cluster that consists of interconnetcted cache servers. A typical implementation consists of HAPROXY and squid servers. The requests from clients are distributed by HAproxy to cache servers. HAproxy use be set to use different algorithms algorithms to maximize the utilization of every server. Round-robin algorithm distributes the load equally to each server. In heterogeneous cluster, weighted round-robin algorithm is used. A weight was assigned to the server based on its processing capabilities. A heterogeneous strucutre of a cluster adds comlexity to the feature enginnring, as we will discuss later.

As CDN providers don't have direct QoS measurement from the clients (the logs from video players, web browser that can show the QoE), so they use the indirect measurement reach ratewhich is calculated from log of the HA proxy of CDN cache groups with a delay about three minutes.

The number of metrics we collected from the cache servers of cache group are different due to the different configuration, from 64 to 134. The features we have constructed are listed in table. There are about 312 features. They include cpu utilization, network utilization, disk utilization, memory utilization.

3. Problem Formulation and Model

We argue that performance evlaluation as a sequence learning problem. Since we are able to collect the caching servers performance metrics and network metrics at a certain time interval, we can use a sequence models to de-

feature list
group 1
cpu1.usage
cpu2.usage
...
mem_cached
mem_buffers_cache_free
memory.swap
disk.used.sda1
disk.used.sda2
...
channeltraffic_in
channeltraffic_in
ioutil_util_sda
ioutil_util_sdb
ioutil_util_sdc
ioutil_util_sdd
...
iowait.wait
hitratio.port8101
hitratio.port8102
...
group 2
aggregate.cpu
aggregate.diskused
aggregated.ioutil
aggregate.CPU.max
....
group 3
aggregate.all_machine.cpu
aggregate.all_machine.diskused
aggregated.all_machine.ioutil
aggregate.all_machine.CPU.max

Table 1: list of candidate input features from one cache serverWe organize the features into groups. The first group is the raw data we directly collected from the caching servers. The second group are the features we construct based on statistics of the metrics of a single machine. The third group are the features we construct based on the statistics of the metrics of the whole cache server groups

scribe relationship between metrics collected from cache groups and reach rate

There are four catogories of sequence learning problem, which are many to one, many to one and many one. Our goal is to model the relationship between collected metrics and reach rate, which is many to one. In general, we can use the following formulation to describe the prediction process.

Giving a sequnce of vectors, $\{x_n\} = \{x_\alpha \in R^d | \alpha \in N\}$, , where d is the number of the features, we use $\{x_n\}$ to represent the sequence and x_t to describe a data point at time t with d dimensions.

Giveng another target sequence, $\{y_n\} = \{y_\alpha \in R | \alpha \in N\}$, our goal is to find the relation between$\{x_n\}$ and $\{y_n\}$, which is

$$y_t = f(x_t, x_{t-1}, ..., x_{t-n+1})$$

where n is the window size and f is the mapping we want to learn from the data.

Many models can be used to approximate f in sequence modeling. The most naive way is to use simple huristics, which is use exponential moving average to linearly map each metrics to a score in a certain time interval. The paremeters are depending on the experience of the operators. This methods are impratical: the model can hardly generalize well and requires tedious repetitive tuning.

3.1. linear models for sequence learning

There are conventional approaches which use data to learn: use the multiple linear regression model(MLR). The MLR method builds a model of a sequence that is composed of a linear part and a random noise part. The linear part models the linear relationship between the dependent variables and independent variables, the random noise reflects the unpredictable randomness. Formally, the model for multiple linear regression, given n observations, is

$$y_t = c + \sum_{i=t-n+1}^{t} (\beta_i x_{t_i}) + \epsilon_t$$

Furthermore, the linear part incorporates historical values of the sequence. Here, y denotes thetarget we want to model, c denotes the constant parameter of the linear decomposition, denotes the parameters to be computed and epsilon reflects the random noise part. The best-fitting line for the observed data is calculated by the least square method.

Linear models are easy to implement and have good interpretation and thus are widely used in many real work. However, linear models are shown not sufficient to describe some nonlinear behaviors of the complex network systems. In many cases, neural networks tend to outperform linear models [A comparison of artificial neural network and time series models for forecasting commodity prices]. In our experiments, we observe a non-linear relationship between the metrics and reach rate.

Deep learning (DL) is a branch of machine learning based on a set of algorithms that attempts to model high-level abstractions in data by using artificial neural network (ANN) architectures composed of multiple non-linear transformations. They have a lot of successful applications in sequnce modeling. Compared to other machine learning techniques, it can detect complex relationships among features,can extract hierarchical level of features from raw data. So it can produce more accurate result and build a model more accurate with less time. They are advantageous for modeling intricate systems because neural networks do not require the user to predefine the feature interactions in the model, which assumes relationships within the data. Instead, the neural network searches for patterns and interactions between features to automatically generate a best fit model.

One of the more popular DL deep neural networks is the Recurrent Neural Network (RNN). RNNs are a class of neural networks that depend on the sequential nature of their input. Such inputs could be text, speech, time series, and anything else in which the occurrence of an element in the sequence is dependent on the elements that appeared before it.

3.2. non-linear for sequence learning

Sequence prediction often involves forecasting the next value in a real valued sequence or outputting a class label for an input sequence.

3.2.1. Basic feed forward neural network

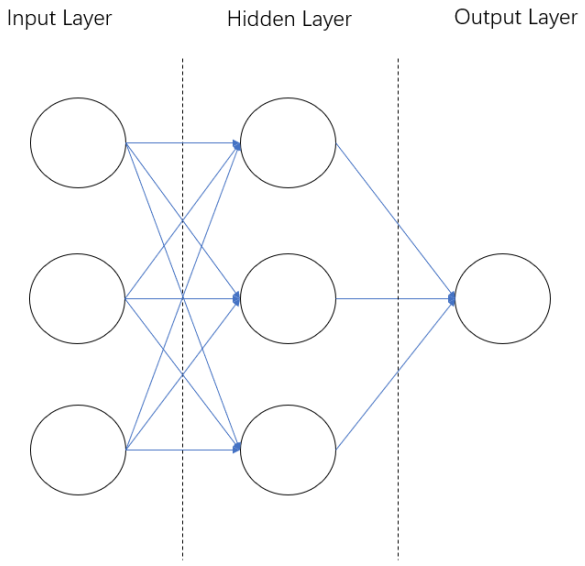

Figure 3: MLP structure

A generic three layered neural network is illustrated in Figure 2. In this study, the input matrix $I \in R^{Mx(NxT+1)}$ where M is the number of training examples and $NxT+1$ is the number of features (metrics that collected from the cache servers) concatenating the bias term. The input matrix is then multiplied by the model parameters matrix W_1 to produce the hidden state matrix h. The Output of the first layer is transformed by an activation function. We can add more layers to the network. interacts with the second parameters matrix Θ^2 to calculate the output O. The size and number of hidden layers can be varied to model systems of varying complexity. The whole process can be imagined as the information propagating foward. The foward propagation is formalized as follows:

$$\begin{aligned} \mathbf{h}_1 &= \text{sigmoid}(\mathbf{W}_1 * \mathbf{I}) \\ \mathbf{h}_t &= \text{sigmoid}(\mathbf{W}_{t-1} * \mathbf{h}_{t-1}) \\ \mathbf{O} &= \text{sigmoid}(\mathbf{W} * \mathbf{h}_{lastlayer}) \end{aligned} \tag{1}$$

where sigmoid is the activation function:

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2}$$

the loss function J use mean square error: J

The process of training any neural network model can be broken down into four steps: (1) Randomly initialize the model parameters, (2) forward propagation algorithm,(3) Compute the cost function J, (4) back propagate using the chain rule (5) using the gradients caculated, adjust weights to minimize the cost function J.

One major assumption for NNs is the independence of samples. For sequence learning problem, however, this assumption doesn't hold true, for the samples of our problem have a temporal relationship: the status of system of the next timestep not only depends on the status in the current timestep but also on the previous timesteps of indefinite length. One solution is to using a sliding window to capture the sequential relationship between the samples. The performance of this method depends on the window size, which isn't pratical for the dependencies length which isn't a fixed value. RNN eliminates the need to find the size of the window [Training and analysing deep recurrent neural networks].

3.2.2. RNN

The promise of recurrent neural networks is that the temporal dependence and contextual information in the input data can be learned[3] [4].

RNNs 3process the input sequence one element at a time and maintain a hidden state vector which acts as a memory for past information. They learn to selectively retain relevant information allowing them to capture dependencies across several time steps. This allows them to utilize both current input and past information while making future predictions. All this is learned by the model automatically without much knowledge of the cycles or time dependencies in data. RNNs obviate the the need for a fixed size time window and can also handle variable length sequences. Moreover, the number of states that can

be represented by an NN is exponential in the number of nodes.

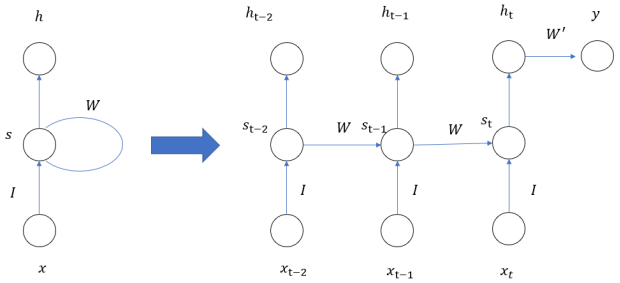

Figure 4: The left shows the backloop structure of RNN and right shows that RNN can be thought as infinite deep layers neural network unfolded along the dimension of time

RNNs maintain a hidden vector $\mathbf{h}$, which is updated at time step t as follows:

$$\mathbf{h}_t = \tanh(\mathbf{W} * \mathbf{h}_{t-1} + \mathbf{I} *_t) \quad (3)$$

where tanh is the hyperbolic tangent function, $\mathbf{W}$ is the recurrent weight matrix and U is a input wight matrix. The hidden state $\mathbf{h}$ is then used to make a prediction

$$\mathbf{y}_t = f(\mathbf{W} * \mathbf{h}_{t-1}) \quad (4)$$

where f can be fully connected layer that linearly maps the hidden state to an output. By using $\mathbf{h}$ as the input to another RNN, we can stack RNNs, creating deeper architectures

$$\mathbf{h}_t^l = \sigma(\mathbf{W} * \mathbf{h}_{t-1}^l + \mathbf{I} * \mathbf{h}_t^{l-1}). \quad (5)$$

The training of RNN use back-propagation through time (BPTT). Training vanilla RNNs is known to be particularly difficult, with vanishing and exploding gradients being one possible explanation [**?**]. The LSTM introduced in the next section address this problems

3.2.3. LSTM

LSTM, introduced in [5], addresses the problem of vanishing gradients by introducing a memory cell. [6] applied in time series. The inner working of LSTM are listed follows:

$$\begin{aligned}
\mathbf{g}_u &= \sigma(\mathbf{W}_u * \mathbf{h}_{t-1} + \mathbf{I}_u *_t) \\
\mathbf{g}_f &= \sigma(\mathbf{W}_f * \mathbf{h}_{t-1} + \mathbf{I}_f *_t) \\
\mathbf{g}_o &= \sigma(\mathbf{W}_o * \mathbf{h}_{t-1} + \mathbf{I}_o *_t) \\
\mathbf{g}_c &= \tanh(\mathbf{W}^c * \mathbf{h}_{t-1} + \mathbf{I}_c *_t) \\
\mathbf{m}_t &= \mathbf{g}_f \odot + \mathbf{g}^u \odot \mathbf{g}^c \\
\mathbf{h}_t &= \tanh(\mathbf{g}_o \odot \mathbf{m}_{t-1})
\end{aligned} \quad (6)$$

here σ is the logistic sigmoid function, $\mathbf{W}_u, \mathbf{W}_f, \mathbf{W}_o, \mathbf{W}_c$ are recurrent weight matrices and $\mathbf{I}_u, \mathbf{I}_f, \mathbf{I}_o, \mathbf{I}_c$ are projection matrices.

3.2.4. LSTM auto-encoder

[4]

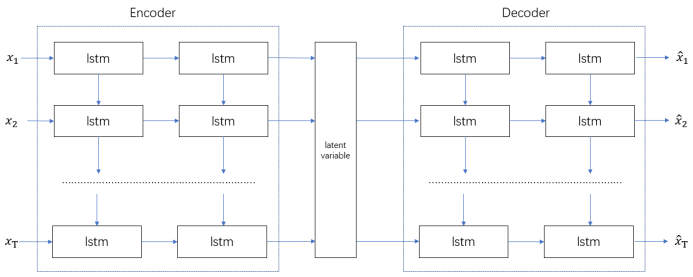

Figure 5: LSTM Auto-encoder Model

An autoencoder contains: an encoding function, a decoding function, and a distance function between the amount of information loss between the compressed representation of your data and the decompressed representation (i.e. a "loss" function). The encoder and decoder will be chosen to be parametric functions (typically neural networks), and to be differentiable with respect to the distance function, so the parameters of the encoding/decoding functions can be optimize to minimize the reconstruction loss, using Stochastic Gradient Descent.

Autoencoders are data-specific. Autoencoders are lossy. Autoencoders are learned automatically from data examples, which is a useful property: it means that it is easy to train specialized instances of the algorithm that will perform well on a specific type of input. It doesn't require any new engineering, just appropriate training data. [7]

4. Methods

4.1. Feature Engineering

The feature enginnering is the process after data-clearning, in which we fill the missing data and reformating it. The propose of this stage is two-fold:

(1) to find a unified equal-length vector representation of all of the cache groups. The metrics collected are in the granuality of caching servers which have differnet dimensionality. As showed in graph. To make things more complex, a cache group may have dfferint number caching servers. As we only care about (2) reduce the dimensionality, for it is very time-consuming to train the models when the number of dimensionality is too high.

4.1.1. Factors analysis

The purpose of factors analysis is: (1) specifying the unit of analysis (2) data summarization and data reduction

As there are hundreds of features, there are many overlaps among the variables. We use correlation in statistis to group highly correlated variables together and create composite measure that can represent each group of variables.

Correlation is an analysis of two or more observed or random variables to determine a dependence between the variables. This dependence can be classified as the probability that changes in one variable affect the behaviour of the second variable. The Pearsons correlation, for instance, defines this dependence in the interval [1.0, 1.0]. Pearsons correlation for two given random variables X and Y is computed by dividing the covariance of both variables with the product of their standard deviations.

Generally, cases of high correlation compute to a value close to 1.0, high anticorrelation is associated with a value close to -1.0 and no correlation is assumed, if the value is around 0.0. In the last case, the variables appear to be independent. Statistical correlation is used in various domains to identify and better understand relationships of variables.

Some Deep Learning algorithms can become prohibitively computationally-expensive when dealing with high-dimensional data.

The correlation matrix we calculatre is pictured in figure 6. The white color denotes the correlation of 1.0 and the black denotes -1.0. The red denotes that there is no linear relationship at all. As we can see from the picture that there are overlaps among the features.

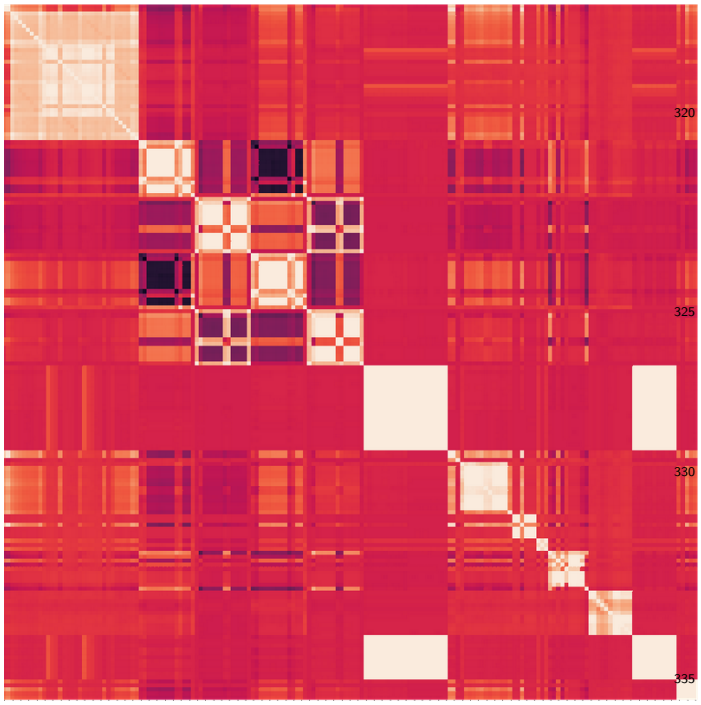

Figure 6: Correlation Matrix of the feature set

After we caculate the correlation matrix for all the features. We use DBSCAN to cluster these features to eliminate duplicated ones.

Algorithm 1 Feature Aggregation and Selection

Input: All features list: F; Neighborhood parameter ϵ;
Output: Selected features list F_S;

Initialize the set of core object: Ω=$\emptyset$;
for *i in F* **do**
  Confirm the ϵ-neighborhood N(i,ϵ);
  if $|N(i,\epsilon)|$>1 **then**
    Add i to the core object set: Ω;
  else
    Add i to the selected features list F_S;
Initialize the number of clusters: k=0;
Initialize the set of features: Γ= Ω;
while $\Omega \neq \emptyset$ **do**
  Check the set of unvisited features: ψ= Γ;
  Random select an core object $o \in \Omega$,initilaize the queue Q=$\langle o \rangle$;
  Γ=$\Gamma \setminus \{o\}$;
  while $Q \neq \emptyset$ **do**
    Fetch an element from Q: q;
    $\Delta = N(q,\epsilon) \bigcap \Gamma$;
    Add the elements in Δ to Q;
    $\Gamma = \Gamma \setminus \Delta$;
  $k = k + 1$;
  Generate the cluster C_k=$\psi \setminus \Gamma$;
  $\Omega = \Omega \setminus C_k$;
for i=1,2...k **do**
  Fetch an element in C_k out to the selected features list F_s
return F_s

4.2. Prediction Model Design

In this section, we introduce how the architecture we use to predict reach rateof CDN cache group using the data ouput from the feature engineering stage.

Prior to fitting the data prediction model, we first conduct a pre-training step to fit an encoder that can extract useful and representative embeddings froma time series. The goals are to ensure that (i) the learned embedding provides useful features for prediction and (ii) unusual input can be captured in the embedded space, which will get further propagated to the prediction network in the next step.

As illustrated in 7Given a multi-variate time series of data, the encoder reads

5. Evaluation

5.1. Experimental Settings and Dataset

Our implementation uses the Google opesouce deep learning library, Tensorflow, version 1.2.0. We ran our experiments on a physical machine running an Ubuntu

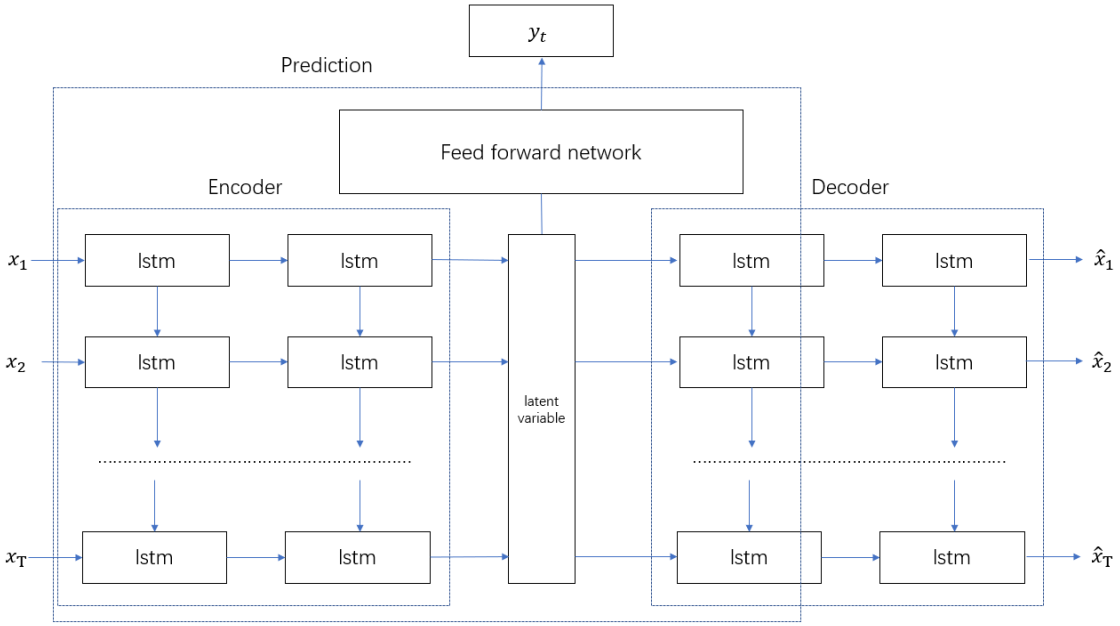

Figure 7: lstm auto-encoder with a deep feadfoward network

16.04 opration system, intel i7-6700HQ, 16 GB memory, and GPU gtx1060.

To test the performance, we select two cache groups with average request above. The first cache servers groups have 13 cache servers while the other have 10 cache servers. The metrics are raw data collected from the cache servers, including cpu utilization, network utilization, disk utilization, memory utilization of two cache groups servers. The frequency of the metrics and reach rateare both minute-by-minute. This data covers the period from November 26, 2016 to January 20 22, 2018, 105 days in total.

In the experiments, 10-fold cross validation is used to evaluate the accuracy of predictions. This divides the entire data set into 10 equal parts. Prediction is repeated 10 times, each time keeping one of the 10 parts as test data and the other 9 parts as training data to build the prediction model. Finally, test results on all the 10 parts are accumulated together to calculate the average prediction accuracy.

In the pre-training procedure We use minibatch stochasti gradient descent (SGD) together with the Adam optimizer [Kingma and Ba, 2014] to train model. The size of the minibatch is 128. The weights can be learned by standard RNN learning algorithm propagation throught with mean squared error as the objective function.

In the training procesure we use a 4 layer feed-foward neural network. We use the batch gradient descent training algorithm to train the neural network.

5.2. Baseline

We compare our model with other baseline model which are listed follow:

1. multiple linear regression model
2. basic three layer feed-foward neural network
3. vanilla LSTM model
4. LSTM encoder-decoder with feed-foward neural network

We use three different evaluation metrics to compare our models and methods. Among them, root mean squared error (RMSE) and mean absolute error (MAE) are two scale-dependent measures, and mean absolute percentage error (MAPE) is a scale-dependent measure. Specifically, assuming $y^{'}$ is the target at time t and y is the predicted value at time t.

$$RMSE(y^{'}, y) = \sqrt{\frac{1}{n} \times \sum_{n=1}^{N} (y^{'}{}_t - y_t)^2} \tag{7}$$

$$MAE(y^{'}, y) = \frac{1}{N} \times \left| y^{'}{}_t - y_t \right| \tag{8}$$

$$MAPE(y^{'}, y) = \frac{1}{N} \times \left| \frac{y^{'}{}_t - y_t}{y^{'}_t} \right| \tag{9}$$

5.3. Performance

We Tuning hypermeters

6. Discussion and Future Work

As there are hundreds cache group in China. Although our method can be used to train on all kinds of cache group of different architecture, the model for one specific cahce group, a unified models for all kinds of cache groups is required. As qualified rate is an indirect measurement of

Table 2: My caption

Models	MAE	MAPE	RMSE
multiple linear regressionmodel	10.0	9.9	8.8
feed-foward neural network with sliding windows	10.0	9.9	8.8
RNN	10.0	9.9	8.8
LSTM encoder-decoder with feed-foward neural network	10.0	9.9	8.8

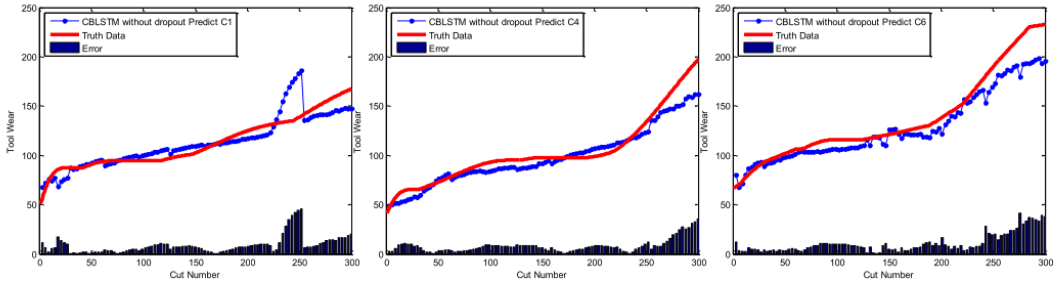

Figure 9. Regression analysis of CBLSTM without dropout for three different testing scenarios including C1, C4 and C6.

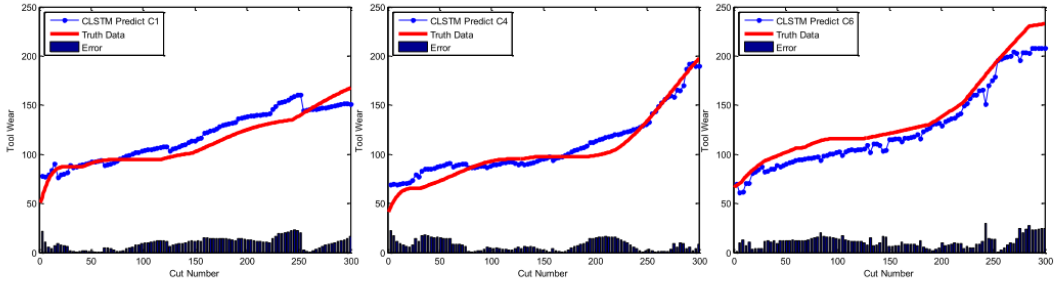

Figure 10. Regression analysis of CLSTM for three different testing scenarios including C1, C4 and C6.

Figure 8: Prediction Results

QoS of clients,collecting data from clients ends will provide useful insights. We also want to develop the online training methods for our models because we observe that the relationship change over time.

7. Related Work

deep-learning; RNN; RNN encoder-decoder; LSTM; LSTM time-series application; sequence learning with lstm:

Deep learning has been widely used in performance modeling.

When evaluating the complex system, the evaluation method can fall into two catogory: model-driven method, data-driven method. In model-drvien method, the mathematical model characterizing the inner components of a system has to be expained explicitly.

However, in data-driven method, the data characterizing the state of system instead of the analytical model is neceessary. From the characteristics of the above methods, the data driven method, which takes the gathered data as basis and is independent of the objects prior knowledge, is a more useful approach for fault prediction and reliability evaluation.

There are three CDN resarch are (1) long-term network planning, including optimized CDN design that relates to PoP selection and cache deployment, and (2) short-term, run-time cache management, including content replacement and prefetching strategies in the CDN network. (3) CDN selection from the clien. Our research falls in the second catogries. Our performance model can provide a decision basis. There are three CDN resarch are (1) long-term network planning, including optimized CDN design that relates to PoP selection and cache deployment, and (2) short-term, run-time cache management, including content replacement and prefetching strategies in the CDN network. (3) CDN selection from the clien. Our research falls in the second catogries. Our performance model can provide a decision basis. There are three CDN resarch are (1) long-term network planning, including optimized CDN design that relates to PoP selection and cache deployment, and (2) short-term, run-time cache management, including content replacement and prefetching strategies in the CDN

network. (3) CDN selection from the clien. Our research falls in the second catogries. Our performance model can provide a decision basis.

8. Conclusion

In this paper we present a deep learning framwork to evaluate the performance of cache group. This paper shows that it is feasible to apply state-of-the-art Deep learning techniques to model networked systems that provides esimation for its performance. The empirical studies shows that it has outputforms the traditional linear method.

References

[1] J. Jiang, S. Sun, V. Sekar, H. Zhang, Pytheas: Enabling Data-Driven Quality of Experience Optimization Using Group-Based Exploration-Exploitation, 14th USENAIX Symposium on NSDI.

[2] H. Mao, R. Netravali, M. Alizadeh, Neural Adaptive Video Streaming with Pensieve, Mohammad Alizadeh MIT Computer Science and Artificial Intelligence Laboratory (2017) 197–210doi:10.1145/3098822.3098843.
URL http://dx.doi.org/10.1145/3098822.3098843

[3] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157–166. doi:10.1109/72.279181.
URL http://www.ncbi.nlm.nih.gov/pubmed/18267787http://ieeexplore.ieee.org/document/279181/

[4] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation.
URL http://www.statnlp.org/wp-content/uploads/2016/02/rnn.pdf

[5] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997) 1735–1780. doi:10.1162/neco.1997.9.8.1735.

[6] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, Long Short Term Memory Networks for Anomaly Detection in Time Series.
URL https://www.researchgate.net/profile/Pankaj_Malhotra3/publication/304782562_Long_Short_Term_Memory_Networks_for_Anomaly_Detection_in_Time_Series/links/5880506308ae71eb5dbfbd10/Long-Short-Term-Memory-Networks-for-Anomaly-Detection-in-Time-Series.pdf

[7] Building Autoencoders in Keras.
URL https://blog.keras.io/building-autoencoders-in-keras.html