A data-driven approach for performance evaluation for cache 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

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 has its drawback. Recently, data-driven paradigm have been shown to greatly outperform traditional methods in many applications, in both accuracy and their quick reactions to the changing environment. We design a framework that using these techniques to build a performance model. Our approach shows an average 6.98% improvement in terms of weighted mean absolute percent error (WMAPE) compared to the baseline models.

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

1. Introduction

There is a trend [1] [2] that using data-driven methods to model complex networked systems. Traditional approach typically simple huristics. These methods have several drawbacks.They cannot accurately reflect and over-simplified 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 models 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 group can be characterised as multi-dimensional, highly non-linear, time variant vecrors. The metrics that collected from members of the CDN cache groups are sequence data that are measured every minute, which 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 prob-

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

lem, 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

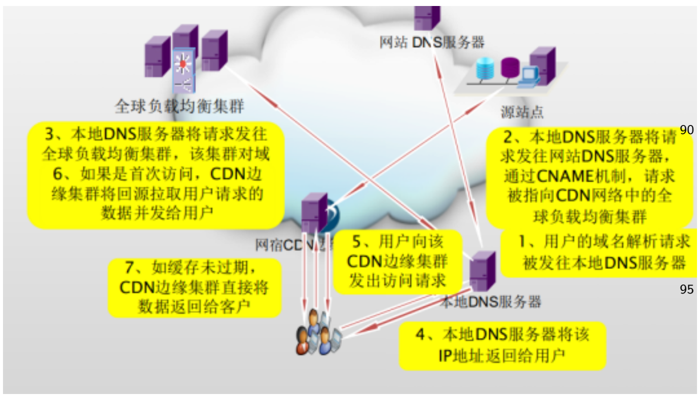

网站 DNS服务器
全球负载均衡集群
源站点
3、本地DNS服务器将请求发往全球负载均衡集群，该集群对域
6、如果是首次访问，CDN边缘集群将回源拉取用户请求的数据并发给用户
2、本地DNS服务器将请求发往网站DNS服务器，通过CNAME机制，请求被指向CDN网络中的全球负载均衡集群
网宿CDN边
5、用户向该CDN边缘集群发出访问请求
1、用户的域名解析请求被发往本地DNS服务器
7、如缓存未过期，CDN边缘集群直接将数据返回给客户
本地DNS服务器
4、本地DNS服务器将该IP地址返回给用户

Figure 1: The figure explains how the content delivery network works:1.2.3.4.5.6

A content delivery network or content distribution network (CDN) 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. [CDNs serve a large portion of the Internet content today, including web objects (text, graphics and scripts), downloadable objects (media files, software, documents), applications (e-commerce, portals), live streaming media, on-demand streaming media, and social networks.]

The architecture of CDN is in figure 2. An CDN cache group is the minimal resource scheduling unit for CDN. A CDN cache group is a load banlanced cluster that consists of interconnetcted cache servers.][https://msdn.microsoft.com/en-us/library/ff648960.aspx] The tasks from clients are distributed requests across multiple servers. Load balancers

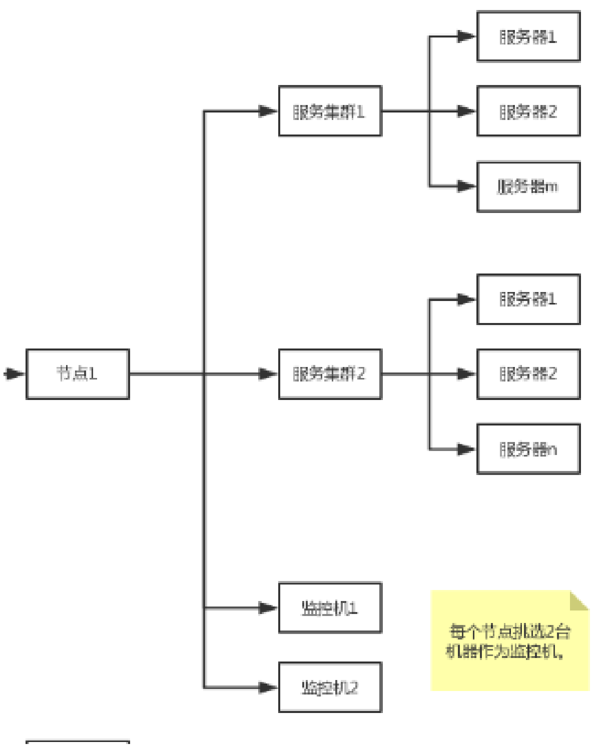


Figure 2: The figure

use different algorithms algorithms to maximize the utilization of every server automatically. 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 cluster adds comlexity to the feature enginnring, as we will discuss later.

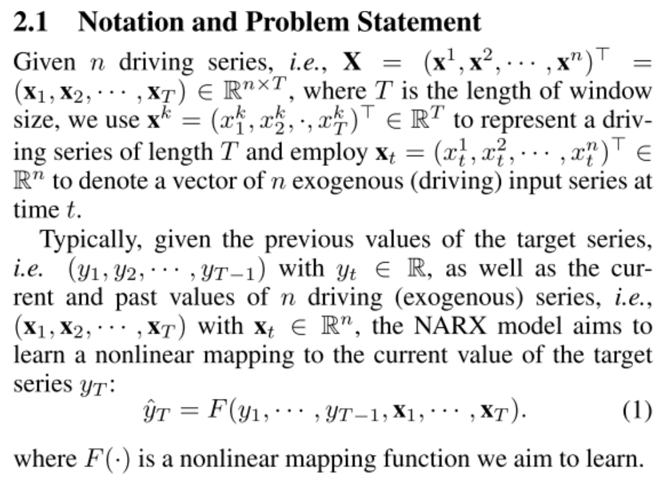
As CDN provicers don't have direct QoS measurement from the clients (the logs from video players, web browser that can show the QoE), so they use the indirect measurement reach ratewhich is collected from log of the HA proxy of CDN cache groups. haproxy =/ ¿ = =/

The data we collected are listed in table. We collected about 130 metrics. It can rouphlyy grouped in the following broad categories: 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 describe relationship between metrics 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. In general, we can use the following formulation to describe the prediction process.



2.1 Notation and Problem Statement

Given n driving series, *i.e.*, $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n)^\top = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T) \in \mathbb{R}^{n \times T}$, where T is the length of window size, we use $\mathbf{x}^k = (x_1^k, x_2^k, \cdot, x_T^k)^\top \in \mathbb{R}^T$ to represent a driving series of length T and employ $\mathbf{x}_t = (x_t^1, x_t^2, \cdots, x_t^n)^\top \in \mathbb{R}^n$ to denote a vector of n exogenous (driving) input series at time t.

Typically, given the previous values of the target series, *i.e.* $(y_1, y_2, \cdots, y_{T-1})$ with $y_t \in \mathbb{R}$, as well as the current and past values of n driving (exogenous) series, *i.e.*, $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ with $\mathbf{x}_t \in \mathbb{R}^n$, the NARX model aims to learn a nonlinear mapping to the current value of the target series y_T:

$$\hat{y}_T = F(y_1, \cdots, y_{T-1}, \mathbf{x}_1, \cdots, \mathbf{x}_T). \tag{1}$$

where $F(\cdot)$ is a nonlinear mapping function we aim to learn.

Figure 3:

The training phase is to learn a best function that minimizes the prediction error as follows:

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

Many models can be used to approximate f in sequence modeling. Conventional approaches use AR models. The AR method builds a model of the time series that is composed of a linear part and a random noise part. Whilst the linear part models the ascertainable chunk of the time series, the random noise reflects the unpredictable randomness in the time series. Furthermore, the linear part incorporates historical values of the time series, which also form the order of the AR model. One commonly refers to the notation AR(l) to indicate how much historical information is used to build the AR model. Equation shows the general autoregressive model for the univariate case. Here, y denotes the time series to be modelled, c denotes the constant parameter of the linear decomposition, denotes the model to be computed and epsilon reflects the random noise part.

$$y_t = c + \sum_{i=1}^{l} (\beta_i y_{t\,i}) + \epsilon_t \tag{2}$$

VAR model is a generalization fo AR models. In VAR models, the relationship between the predictor and the target variable is simply described using a linear model as follows: It is used to find a linear model that incorporates the influence of multiple time series into the actual value of a target variable yt.

As such, the equation contains the vectors Yt, Ytl, the values ?t, c and the model matrices Al. The vectors Yt and Ytl are composed of all incorporated variables (x, y, z, ...) at time stamp t and t l. Hence, the i-th element in these vectors is the i-th incorporated variable at the respective time stamps. The matrices A1 . . .Al reflect the models that need to be fitted. The VAR model is shown in equation 2.21.

$$equationofVAR \tag{3}$$

Linear models are easy to implement and have good interpretation and thus are widely used in many real work time series analysis problems. 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 AR-based models [A comparison of artificial neural network and time series models for forecasting commodity prices]. We use deep learning as alternaives.

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) ()Figure) architectures composed of multiple non-linear transformations. They have a lot of successful applications in speech and image recognition, machine translation, and forecasting of financial time series. 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 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.

The essence of DL is to compute hierarchical features or representations of obser-vational data, where the higher-level features or factors are defined from primary lower-level measurements. Based on the features extracted from the data in the training set, the calculations within the model are adjusted so that known inputs produce desired outputs

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.

3.1. deep learning in sequence learning

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

3.1.1. Basic feed forward neural network

A generic three layered neural network is illustrated in Figure 2. In this study, the input matrix is an x (m x n) array where is the number of training examples and is the number of features (DC input variables), including the IT load, weather conditions, number of chillers and cooling towers running, equipment setpoints, etc. The input matrix is then multiplied by the model parameters matrix x 1 to produce the hidden state matrix [6]. In practice, acts as an intermediary state that interacts with the second a a parameters matrix 2 to calculate the output h (x) [6]. The size and number of hidden layers can be varied to model systems of varying complexity.

The process of training a neural network model can be broken down into four steps, each of which are covered in greater detail below: (1) Randomly initialize the model parameters , (2) Implement the forward propagation algorithm, (3) Compute the cost function J() , (4) Implement the back propagation algorithm and (5) Repeat steps 2 to 4 until convergence or the desired number of iterations.

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.

3.1.2. 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 process 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. 11

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

$$\mathbf{h}_t = \tanh(\mathbf{W} * \mathbf{h}_{t-1} + \mathbf{I} *_t) \tag{4}$$

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

$$\mathbf{y}_t = \text{softmax}(\mathbf{W} * \mathbf{h}_{t-1}) \tag{5}$$

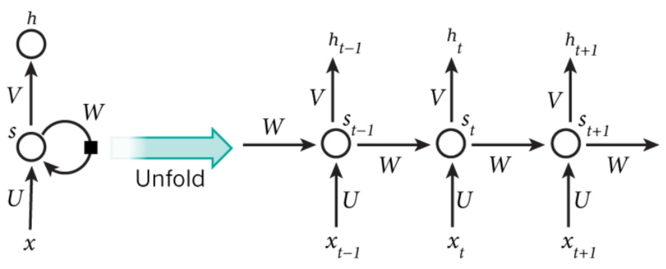


Figure 4: RNN Architecture

where *softmax* provides a normalized probability distribution over the possible classes and $\mathbf{W}$ is a weight matrix. By using $\mathbf{h}$ as the input to another RNN, we can stack RNNs, creating deeper architectures [**?**]

$$\mathbf{h}_t^l = \sigma(\mathbf{W} * \mathbf{h}_{t-1}^l + \mathbf{I} * \mathbf{h}_t^{l-1}). \tag{6}$$

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

3.1.3. LSTM

LSTM, introduced in [5], addresses the problem of vanishing gradients by introducing a memory cell which ensures constant error flow and gating units. [6] applied in time series. The inner working of LSTM are listed follows:

$$\begin{aligned}
\mathbf{g}^u &= \sigma(\mathbf{W}^u * \mathbf{h}_{t-1} + \mathbf{I}^u *_t) \\
\mathbf{g}^f &= \sigma(\mathbf{W}^f * \mathbf{h}_{t-1} + \mathbf{I}^f *_t) \\
\mathbf{g}^o &= \sigma({}_u sage_c pu1thbfW^o * \mathbf{h}_{t-1} + \mathbf{I}^o *_t) \\
\mathbf{g}^c &= \tanh(\mathbf{W}^c * \mathbf{h}_{t-1} + \mathbf{I}^c *_t) \\
\mathbf{m}_t &= \mathbf{g}^f \odot + \mathbf{g}^u \odot \mathbf{g}^c \\
\mathbf{h}_t &= \tanh(\mathbf{g}^o \odot \mathbf{m}_{t-1})
\end{aligned} \tag{7}$$

3.1.4. RNN encoder-decoder

[4] applications: machine translation, learning to excute, image captioning, conversational modeling

RNN Encoder-Decoder, consists of two recurrent neural networks (RNN) that act as an encoder and a decoder pair. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable-length target sequence. [4] also known as sequence embedding. The point of training an autoencoder is to make an RNN learn how to compress a relatively long sequence into a limited, dense vector.

3.2. lstm auto-encoder

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

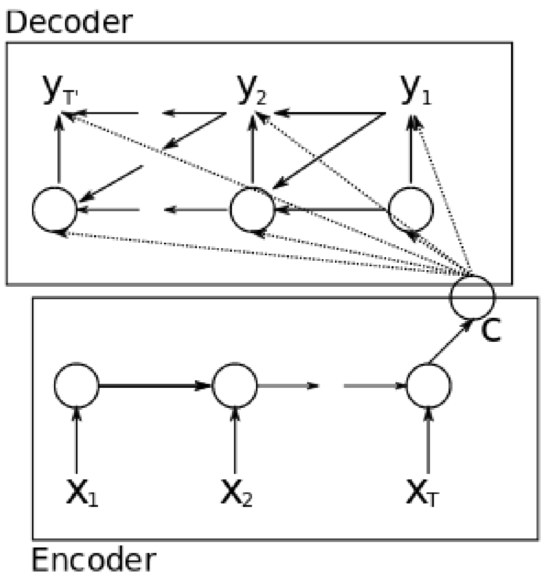


Figure 5: neural network architecture

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]

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.

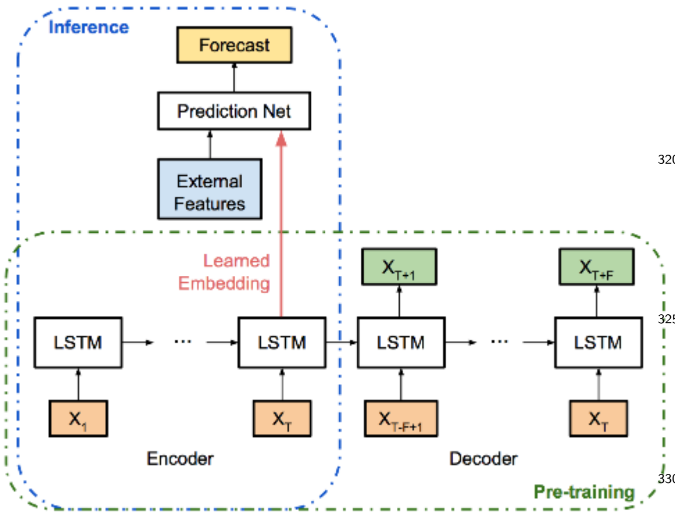


Figure 6: neural network architecture

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 smurrazation data reduction (3) variable selection As there are hundreds of variables, 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 (except in the special case of auto-correlation 2.2.1) to deter- mine 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.

$$\mathbf{cor}_p = p_{X,Y} = \frac{cov(X,Y)}{XY} \tag{8}$$

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. Examples are: medicine, biology, psychology or astronomy

Feature selection: The benefit of this feature selection process is two-fold: (a) a reduced feature set will control the model complexity during model learning, and (b) the processes gain more insights on the complex interaction of different matrics. [8]. Some Deep Learning algorithms can become prohibitively computationally-expensive when dealing with high-dimensional data.

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.

feature	meaning
cpu1.usage	cpu used ratio of cpu1
cpu2.usage	cpu used ratio of cpu2
...	
mem_cached	the size of memory cache
mem_buffers_cache_free	the size of memory cache buffer
memory.swap	the size of memory swap
disk.used.sda1	the size of disk used in sda1
disk.used.sda2	the size of disk used in sda2
...	
channeltraffic_in	
channeltraffic_in	
ioutil_util_sda	
ioutil_util_sdb	
ioutil_util_sdc	
ioutil_util_sdd	
...	
iowait.wait	
hitratio.port8101	
hitratio.port8102	
...	
aggregate.cpi	the cpu used ratio of all cpu
aggregate.diskused	the disk used form all disk
aggregated.ioutil	the overall ioutil
aggregate.CPU.max	the max cpu used ratio from all cpu
....	
aggregate.all_machine.cpi	the cpu used ratio of the whole group
aggregate.all_machine.diskused	the disk used form all disk
aggregated.all_machine.ioutil	the overall ioutil
aggregate.all_machine.CPU.max	the max cpu used ratio from all cpu

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 is the feature we caculate based on the first group. The

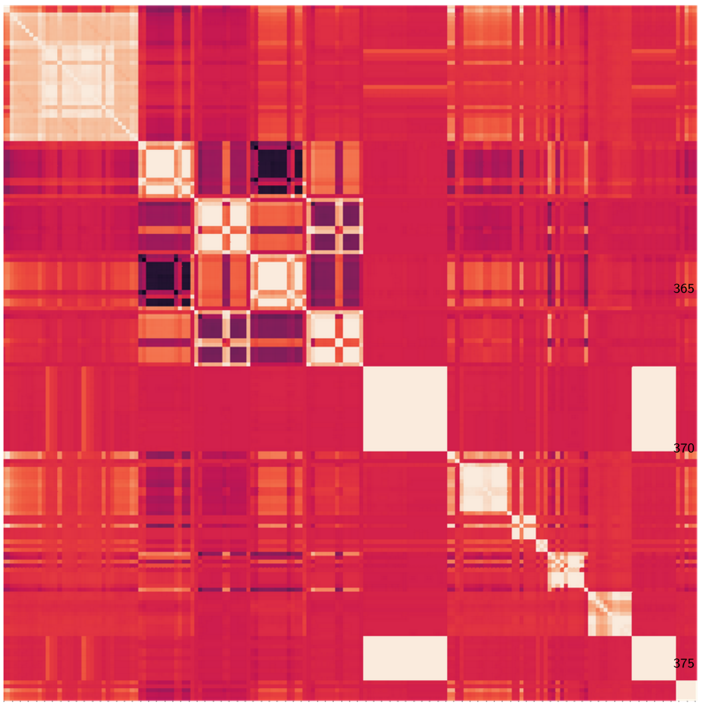


Figure 7: Correlation Matric

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.

lstm encoder-decoder:

prediction networks

5. Evaluation

5.1. Experimental Settings

Our implementation using the Google machine learning library, Tensorflow, version 1.2.0. We ran our experiments on a physical machine running an Ubuntu 16.04 imag, interl i7, 8 GB memory, and GPU gtx1060. The data is collected from the daily operation of CDN cache groups In this section, we illustrate the model performance using reach ratedata collected from a cache group in a selected PoP that provide content caching service for several customers.

In the experiments, 10-fold cross validation is used to eval- uate 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.

Table 2: My caption

location	Persistent	LSTM	LSTM encoder-decoder	Our
Shanghai	10.0	9.9	8.8	7.7
Shenzhen	10.0	9.9	8.8	7.7
Zhejiang	10.0	9.9	8.8	7.7

Training procedure We use minibatch stochastic gradient descent (SGD) together with the Adam optimizer [Kingma and Ba, 2014] to train the model. The size of the minibatch is 128. the parameters can be learned by standard back propaga- tion with mean squared error as the objective function:

Loss=

5.2. Baseline

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

1. VAR Model: a linear model that takes last ouput as the prediction
2. basic Three layer neural network
3. Vanilla LSTM Model
4. LSTM encoder-decoder with multiple-layers perceptions
5. LSTM encoder-decoder with SVM

To measure the effectiveness of various methods for prediction, we consider three different evaluation metrics. Among them, root mean squared error (RMSE) [Plu- towski et al., 1996] and mean absolute error (MAE) are two scale-dependent measures, and mean absolute percentage error (MAPE) is a scale-dependent measure. Specifically, assuming yt is the target at time t and yt is the predicted value at time t, RMSE is defined as RSME= and MAE is denoted as MAE=. When comparing the prediction performance across different datasets, mean absolute percentage error is popular because it measures the prediction deviation proportion in terms of the true values, i.e, MAPE=.

5.3. Performance

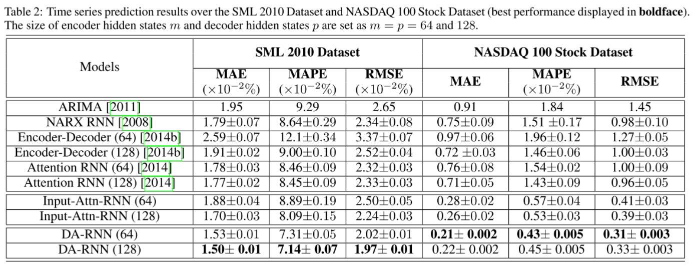


Table 2: Time series prediction results over the SML 2010 Dataset and NASDAQ 100 Stock Dataset (best performance displayed in **boldface**). The size of encoder hidden states m and decoder hidden states p are set as $m = p = 64$ and 128.

Models	SML 2010 Dataset			NASDAQ 100 Stock Dataset		
	MAE ($\times 10^{-2}\%$)	**MAPE** ($\times 10^{-2}\%$)	**RMSE** ($\times 10^{-2}\%$)	**MAE**	**MAPE** ($\times 10^{-2}\%$)	**RMSE**
ARIMA [2011]	1.95	9.29	2.65	0.91	1.84	1.45
NARX RNN [2008]	1.79±0.07	8.64±0.29	2.34±0.08	0.75±0.09	1.51 ±0.17	0.98±0.10
Encoder-Decoder (64) [2014b]	2.59±0.07	12.1±0.34	3.37±0.07	0.97±0.06	1.96±0.12	1.27±0.05
Encoder-Decoder (128) [2014b]	1.91±0.02	9.00±0.10	2.52±0.04	0.72 ±0.03	1.46±0.06	1.00±0.03
Attention RNN (64) [2014]	1.78±0.03	8.46±0.09	2.32±0.03	0.76±0.08	1.54±0.02	1.00±0.09
Attention RNN (128) [2014]	1.77±0.02	8.45±0.09	2.33±0.03	0.71±0.05	1.43±0.09	0.96±0.05
Input-Attn-RNN (64)	1.88±0.04	8.89±0.19	2.50±0.05	0.28±0.02	0.57±0.04	0.41±0.03
Input-Attn-RNN (128)	1.70±0.03	8.09±0.15	2.24±0.03	0.26±0.02	0.53±0.03	0.39±0.03
DA-RNN (64)	1.53±0.01	7.31±0.05	2.02±0.01	**0.21± 0.002**	**0.43± 0.005**	**0.31± 0.003**
DA-RNN (128)	**1.50± 0.01**	**7.14± 0.07**	**1.97± 0.01**	0.22± 0.002	0.45± 0.005	0.33± 0.003

Figure 8: Prediction Results

compare four different models in terms of training time and accuracy

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 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

When evaluating the's complex system, the evaluation method can fall into three catogory: model-driven method, data-driven method. In model-drvien method, the mathematical model characterizing the inner components of a system has to be all known$_{C}DN$.

The data characterizing the state of system instead of the analytical model is neceessary. [?].[2].[?]. [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]l

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 [9]. (3) CDN selection from the client[1] [?]. Our research falls in the second catogries. Our performance model can provide a decision basis

deep-learning; RNN; RNN encoder-decoder; LSTM; LSTM time-series application; LSTM with attention; sequence learning with lstm: Real-Time Prediction of Taxi Demand Using Recurrent Neural Networks

Deep learning has been widely used in performance modeling.

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-Se
pdf

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

[8] J.-S. Yeom, J. J. Thiagarajan, A. Bhatele, G. Bronevetsky, T. Kolev, Data-Driven Performance Modeling of Linear Solvers for Sparse Matrices, in: 2016 7th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), IEEE, 2016, pp. 32–42. doi:10.1109/PMBS.2016.009.
URL http://ieeexplore.ieee.org/document/7836412/

[9] G. Tang, K. Wu, R. Brunner, Rethinking cdn design with distributee time-varying traffic demands, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9. doi:10.1109/INFOCOM.2017.8057028.