Future Generation Computer Systems 90 (2019) 477–488

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Bigdata logs analysis based on seq2seq networks for cognitive Internet of Things

Pin Wu [a], Zhihui Lu [b,*], Quan Zhou [a], Zhidan Lei [a], Xiaoqiang Li [a], Meikang Qiu [c], Patrick C.K. Hung [d]

[a] School of Computer Engineering and Science, Shanghai University, Shanghai, China
[b] School of Computer Science, Fudan University, Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, Shanghai, China
[c] Department of Computer Science, Pace University, New York, NY 10038, USA
[d] Faculty of Business and IT, University of Ontario Institute of Technology, Canada

HIGHLIGHTS

- According to the characteristics of bigdata, this paper introduces a recurrent neural network based bigdata log analysis method used to predict the most likely future events.
- In this work we elaborate the use of distributed ways to collect and store bigdata logs, event-location and vectorized representations of logs.
- A log fusion algorithm is introduced to convert the logs of each component of bigdata into structured data by removing noise data, adding timestamps and classification labels.
- A sequence-to-sequence algorithm is improved by using an attention mechanism to balance deviations in the sequence and an adjustor to globally fit the data of output data.
- Compared with the previous predictive method, the RMSE is reduced by 46.65% and the R2 fitting degree is improved by 14.28%.

ARTICLE INFO

Article history:
Received 16 April 2018
Received in revised form 6 August 2018
Accepted 11 August 2018
Available online 20 August 2018

Keywords:
Cognitive computing
Internet of Things
Bigdata
Recurrent neural network
Log analysis

ABSTRACT

While bigdata system processes high-volume data at high speed, it also generates a large amount of logs. However, it is hard for people to predict future events based on massive, multi-source, heterogeneous bigdata logs. This paper proposes a comprehensive method for smart computation and prediction of massive logs in the internet of things (IoT). Traditional machine learning, Hidden Markov Model (HMM) and Autoregressive Integrated Moving Average Model (ARIMA) methods are not accurate enough to predict time series based data over time. In this work we first elaborate the distributed collection and storage, event location, and vectorized representations of bigdata logs. Next, we present a log fusion algorithm to convert the logs (unstructured text data) of each component of bigdata into structured data by removing noise, adding timestamps and classification labels. Then, we introduce a predictive model for bigdata system. We use an attention mechanism to improve sequence to sequence (seq2seq) algorithm and add an adjustor to globally fit the data distribution. Our experimental results show that the neural network model trained by our method has a good performance with the real-world data. Compared with the previous predictive method, the root mean square error (RMSE) is reduced by 46.65% and the R-squared (R2) fitting degree is improved by 14.28%.

1. Introduction

According to BIIntelligence's forecast report [1], there will be nearly 22.5 billion Internet of Things devices by 2021, nearly 16 billion more than those in 2017. With the burst of data from those devices, it is a challenge to improve performance and achieve intelligence using the current IoT integrated with cognitive and cooperative mechanisms. Fortunately, bigdata and deep learning technologies have been developed rapidly in recent years. Now bigdata technology can provide fast computing and massive storage through parallel computing and distributed storage system. IoT can take advantage of these technologies to expand horizontal scalability. However, in the process of providing bigdata services, a huge amount of logs are generated to record the behavior of

* Corresponding author.
E-mail addresses: wupin@shu.edu.cn (P. Wu), lzh@fudan.edu.cn (Z. Lu), quanzhou10@fudan.edu.cn (Q. Zhou), sapphireice@shu.edu.cn (Z. Lei), xqli@shu.edu.cn (X. Li), mqiu@pace.edu (M. Qiu), patrick.hung@uoit.ca (P.C.K. Hung).

https://doi.org/10.1016/j.future.2018.08.021
0167-739X/

bigdata systems at runtime. It is necessary to monitor and analyze these logs to better guarantee the functionality of bigdata services. Analyzing and processing these massive logs has gradually become a necessary component for securing bigdata services. Bigdata systems often use multiple components to accomplish different tasks. These components include Hadoop HDFS, MapReduce, Hbase, Hive, Spark, Redis, Kafka, MPP, Zookeeper, YARN and others. These components generate their own log respectively during operation. Some of these heterogeneous logs are independent of each other, and some are interdependent. In general, bigdata system log has notable features, such as massive, multi-source, heterogeneous. And thus it is a challenge for people to analyze and predict them.

Due to the complex structures of data, it is very complicated to extract the characteristics of the system log manually. People want to let machines extract knowledge in an automated way. And based on their previous and growing logs, these machines can create new information for specific and unforeseen scenarios [2]. We need to find a way that can smartly compute, analyze and process these massive log data without large-scale human and material resources.

Obviously, the first task among of log analysis is to collect the logs. Due to the large number of nodes in the bigdata system and the large number of components, a distributed collection technology and a mass storage method are required. In addition, if we simply collect those logs, we can only perform basic query and statistical analysis tasks. However, managers will not have time to carefully read the massive daily log information. Therefore, it is significant to mine the valuable and simple information from massive logs. Nassar et al. [3] have found that it is feasible to predict system failures based on error logs. He et al. [4] reviewed and evaluated six log-based anomaly detection methods. Thus, if we can further study the log using algorithms such as the data collection, data mining and machine learning, we can conduct a higher-level semantic association analysis, and even predict the future events.

Log analysis is an important area of research and many research works have been conducted for different applications [5]. Log analysis can be used for debugging programs, troubleshooting problems, ensuring system health, avoiding vulnerabilities, predicting possible attacks and maximizing advertising revenue. For example, when a function of a bigdata system suddenly slows down, we are often less likely to find obvious error messages. Even if we can quickly find the cause of the error, and promptly handle this error, the loss has been caused. So, we need to be able to accurately predict the future events, which is both an exciting and challenging goal.

Since the logs are generated in chronological order, people began to analyze them using time series based techniques. It is very meaningful to mine the time series data quickly and effectively. People are always trying hard to find the relevant method. Although great progress has been made, there is still room for improvement in terms of complexity and accuracy. Recently, researchers have made great breakthroughs in intelligent data analysis by neural networks. Inspired by these works, this paper proposes a new method which uses the recurrent neural network of seq2seq to learn the logs, and generate the prediction model. Massive logs generated from different components are input to the model in real time. Then, we can do data analysis such as predicting the possible problems of other components, generating the most possible log sequences and providing parameters for the resource scheduling process. Once smart and reliable log analysis methods are available, managers can better adjust node plans for IoT systems in advance. We develop a seq2seq-based predictive analysis method to support better management of IoT system to meet the requirement of efficient and accurate analysis of logs. The experimental results show that the proposed method is feasible to compute, analyze and process those massive logs smartly with better performances than previous systems.

Log analysis plays a key role in the operation, maintenance, and optimization of IoT systems. This paper has been extended from previous works on log analysis. Our works use the latest technology to analyze and predict the IoT logs. This paper covers our method, model, real-world experimental results and experiences. The main points of this paper are described in the following sections.

- describe the development process and history of log analysis and analyze its development trends and challenges. (Section 2)
- introduce a seq2seq neural network model to analyze and predict logs and formalize our proposed methodological model. (Section 3)
- introduce a log-based attention mechanism and a novel adjustor for spatial conversion embedded in a seq2seq neural network to improve prediction accuracy (Section 4)
- present experiments with real-world data in comparison with HMM, ARIMA, and mainstream machine learning algorithms. Then, analyze and discuss the experimental results. (Section 5)

2. Related work

We are going to review what researchers have done on log analysis and neural networks at first. Based on these successful practices, we apply seq2seq network to the analysis of bigdata logs. Most previous log analysis studies assumed a single event process and then statistically fitted the log data. Ting-Ting Y. Lin [6] suggested that it would be better to combine multiple events and classified error log event into two types of processes: transient and intermittent. Based on the shape of the interarrival time function of the intermittent errors observed from actual error logs, Lin developed a failure prediction heuristic method named the Dispersion Frame Technique (DFT). Lin showed the DFT can achieve a high accurate rate in failure prediction.

After that, R Ivancsy et al. [7] started to mine the frequency patterns in the log, discovering hidden information from web log data. The goal of discovering frequent patterns in web log data is to obtain information about the navigational behavior of the users. These patterns were expressed in the form of association rules. However, this traditional statistical approach does not provide enough contextual information. Thus, for time series prediction, this method is limited.

With the advance of various application systems, multi-source heterogeneous log analysis has become a research hotspot. People began to study heterogeneous log from different aspects [8]. Among the typical algorithms, Asif-Iqba [9] proposed a clustering algorithm to filter redundant log events and then aggregate the log events. Traditional log analysis was often in accordance with the established sequence of rules on the log sequence to determine the specific event. In fact, this method can extract features from the logs, but it can only be adapted to a specific environment. Recently, Xu Yang proposed a hybrid artificial immune algorithm based on mixed log to deal with heterogeneous log. He et al. [10] designed and implement a parallel log parser on top of Spark, a large-scale data processing platform to normalize bigdata logs.

The emergence of massive logs brings the problem of processing efficiency. D Logothetis et al. [11] presented an in-situ MapReduce architecture that mined data on location, bypassing the cost and wait time of this store-first-query-later approach and improved the ability to accommodate increasing log generation rates. Yu and wang [12] proposed a mass log data processing and data mining methods based on Hadoop to achieve scalability and performance. Then, H Hingave et al. [13] proposed a log analyzer with the combination of Hadoop and MapReduce paradigm. The combination of

Hadoop and MapReduce programming tools made it possible to provide batch analysis in minimum response time and in-memory computing capacity in order to process log in a highly available, efficient and stable way. However, these methods are not good to analyze the relationship between the timing and context of the log and dig out enough valuable information. Dewangan et al. [14] showed a distributed framework in combination of Hadoop and MapReduce to analyze event log file.

RA Epperly et al. [15] presented an overview of the technology that infrared thermography can help to proactively identify equipment failures before they can occur. People began to predict future events by using regression models based on machine learning. Rich Caruana [16] focused on the machine learning process and covered two classes of algorithms for solving function approximation problems: penalized linear regression methods and ensemble methods. R Caruana [17] evaluated performance on three metrics: accuracy, area under curve(AUC), and squared loss. He studied the effect of increasing dimensionality on the performance of the learning algorithms and presented methods for building identifying fitting and checking models for time series and dynamic systems. Brooks et al. [18] presented a novel method for converting educational log data into features suitable for building predictive models of student success. Unlike cognitive modeling or content analysis approaches, these models were built from a time series interactions between learners and resources.

Time series is a very common data structure and people have always been very interested in studying it. People began to analyze and predict them by using time series based algorithms such as HMM [19], ARIMA [20] and ANN [21] in this case. However, Hidden Markov Model(HMM) is a time series probabilistic model that describes process states using a single discrete random variable, and does not perform well when there is not enough prior knowledge about the given series. The basic idea of the ARIMA model is that the data sequence formed by the predicted object over time is considered as a random sequence, and a certain mathematical model is used to approximately describe the sequence. Once this model is identified, future values can be predicted from the past and present values of the time series. Modern statistical methods and econometric models have been able to help us to predict the future to some extent. ARIMA is actually a linear regression, but it just focuses on the regression of its own historical data. It is suitable for the data that has a high and stable correlation.

Recurrent neural networks (RNNs) originated from Hopfield Networks proposed by Saratha Sathasivam. Since 1986, the network was replaced by fully connected neural networks and some traditional machine learning algorithms. However, the method of fully connected neural network requires massive parameters which greatly increase the computational load, and even worse, it cannot acquire the time series information in the data. The appearance of RNN mutations become very good for processing and predicting sequence data [22]. In 1997, Sepp Hochreiter and Jurgen Schmidhuber [23] proposed that Long Short Term Memory (LSTM) solves the problem of gradient disappearance of the earliest recurrent neural networks. The algorithm has three gates. The addition of a forgotten gate is an idea. This gate will selectively forget the insignificant information to solve the problem. Since then, people have been making various changes to LSTM to adapt to different kinds of tasks [24,25]. This article is also based on the algorithm designed to predict the massive, multi-source, heterogeneous logs.

Le et al. [26] presented a general end-to-end approach to sequence learning that made minimal assumptions on the sequence structure. In 2014, K Cho et al. [27] proposed a novel neural network model called RNN Encoder–Decoder that consists of two recurrent neural networks to improve the performance of a statistical machine translation system. Then, Dzmitry Bahdanau et al. [28] propose to allow a model to automatically search for parts of a source sentence that are relevant to predicting a target word, without encoding a source sentence into a fixed-length vector. People are constantly applying seq2seq to different fields and have achieved very good results.

The intelligent prediction of logs is also very meaningful for IoT. With the development of artificial intelligence, machines are becoming more and more intelligent. In terms of computing power, machines have gone far beyond humans. But in terms of cognitive ability, the machine is still in its infancy, even less than a child of one or two years old. To improve intelligence of the system, cognitive computing is a key point. AI pioneer Allen Newell [29] thought that our ultimate goal was a unified theory of human cognition. Historically, many disparate fields have taken radically different approaches to explore mind-like computation [30]. IoT devices generate and exchange more data than before. Extracting the useful information from these data becomes a hectic task. Consequently, many techniques have been proposed to analyze large amounts of logs and enhance decision-making [31]. People gradually let the machine gain more cognitive ability. Although the researchers have done a lot of works on log prediction, they have not been able to fully utilize deep learning to further enhance the results. Next we start to introduce our proposed seq2seq method.

3. Bigdata logs predictive analysis model

Before we introduce our method, let us describe a few definitions in this section. This section consists of sequence model definition (Section 3.1), seq2seq model definition (Section 3.2), attention mechanism definition (Section 3.3), history window and predictive window definition (Section 3.4), adjustment factor definition (Section 3.5) and log pattern definition (Section 3.6).

3.1. Sequence model definition

Let $l_1, l_2, l_3, \ldots, l_n$ be the time series data with history window H and predicting window P. Then it can be expressed as

$$l_{T-H}, l_{T-H+1}, \ldots, l_T, l_{T+1}, l_{T+2}, \ldots, l_{T+P} \tag{1}$$

where n is the length of the sequence. Then, its probability can be expressed as

$$p(S) = p(l_1)p(l_2|l_1), \ldots, p(l_n|l_1, l_2, \ldots, l_{n-1}) \tag{2}$$

In Eq. (2), we need to calculate the conditional probability n−1 times. When the value of n is large, it will be very computationally expensive. Here we need to make a trade-off between computational accuracy and complexity without causing large deviations. We can sacrifice a little bit probability accuracy in exchange for computational efficiency. Then, we assume that the probability of the current log is only relevant to the previous n−1 log events. Then sequence S's probability can be expressed as

$$p(S) \approx p(l_n|l_1, l_2, \ldots, l_{n-1}) \tag{3}$$

The model proposed in this paper can simulate the function of maximum likelihood estimation (MLE). Sometimes the probability of sequence S can be calculated by

$$p(S) \approx \frac{Count(l_1, l_2, \ldots, l_n)}{Count(l_1, l_2, \ldots, l_{n-1})}, \tag{4}$$

where Count(S) indicates the number of log sequence S appearing in the training log set. The larger the number of the training logs, the more reliable the result of the parameter estimation. Similar to Le et al.'s works [27,32], we trained a model by maximizing the probability of a correct target information T when given the source log sequence S. And our goal is

$$max\{\frac{1}{|S|}\sum_{T,S\in S}\log p(T|S)\}, \tag{5}$$

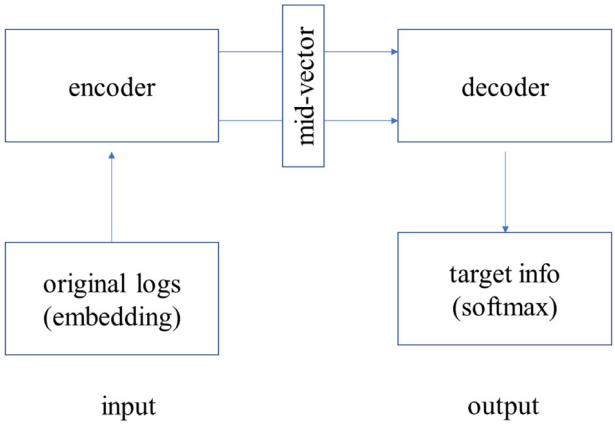


Fig. 1. Predictive model framework.

where S is log sequence from the training set. Once training is complete, we can make predictions by getting the most likely sequences according to trained model:

$$\hat{T} = argmax_T p(T|S) \tag{6}$$

3.2. Seq2seq model definition

The Seq2seq network consists of two layers(encoding layer and decoding layer). The encoding layer is a dynamic multi-layer LSTM. The encoding layer actually acts as the function of neural network memory. The log vectors are sequentially input into the network, and the network will output a result with the context information of the sequence. It only retains the last hidden state, which is equivalent to condensing the entire sequence together and storing it as a content vector for later decoding layer. The decoding layer can be divided into two parts: training and inference. The decoding layer converts the hidden vector into the target sequence.

The inputs and outputs of training and predictive models consist of a sequence of vectors, which are called log2vectors. First, we convert multi-source heterogeneous log data into vectors of the same dimension and store them into a dictionary. Then these vectors are input to the encoding layer based on the chronological order of the log data (training set). These input vectors are converted into hidden layer state vectors by the encoding layer. Then the target information is generated by the decoding layer. The target information for this output is also a sequence, which is artificially added with a start character ⟨*START*⟩ and an end character ⟨*END*⟩ (see Fig. 1).

Encoding layer and decoding layer are mainly composed of neural units, shown in Fig. 2. Let x_t be the input at time t and s_t in Eq. (7) be the state information, which can also be thought of the network memory. y_t is the output at time t, U, V and W are three weight matrices shared by the whole network. s_t can be calculated by

$$s_t = f(Ux_t + Ws_{t-1}) \tag{7}$$

3.3. Attention mechanism definition

We will lose part of the information in the encoding stage. Once the long-term sequence information is lost, it will seriously affect the result of decoding. The most recent logs tend to have more influence over the hidden state vector. Therefore, we introduce attention mechanism to solve this problem. Attention mechanism is proposed in computer vision. Normally, we want the areas in the image that need attention to have high resolution, while background information should be low resolution. [33]. For the same reason, we would like to pay more attention to the important logs and less attention to unimportant logs. Recurrent sequence generators have recently achieved very good results in a variety of tasks including machine translation, handwriting [28,32] and image title generation [34]. After introducing the attention mechanism into those models, they often got better results. So, we borrowed this idea to improve the seq2seq model.

As shown in Fig. 3, the mapping relationship between the input sequence x1, x2, x3 and the output sequence y1, y2, y3 is many-to-many. The correspondence between them is an attention weight matrix. The vector output from the encoding layer is multiplied by this weight matrix to obtain the hidden layer state vector(middle-vector). In other words, the vector subject to a spatial transformation operation. In this way, the decoding layer will decode the middle-vector according to the weight. As a result, logs that have greater weight would have a greater impact on the output of the decoding layer.

3.4. History window and predicting window definition

The history window is the number of events that happened from the past to the present. The predicting window is the number of events that will happen from the present to future, as shown

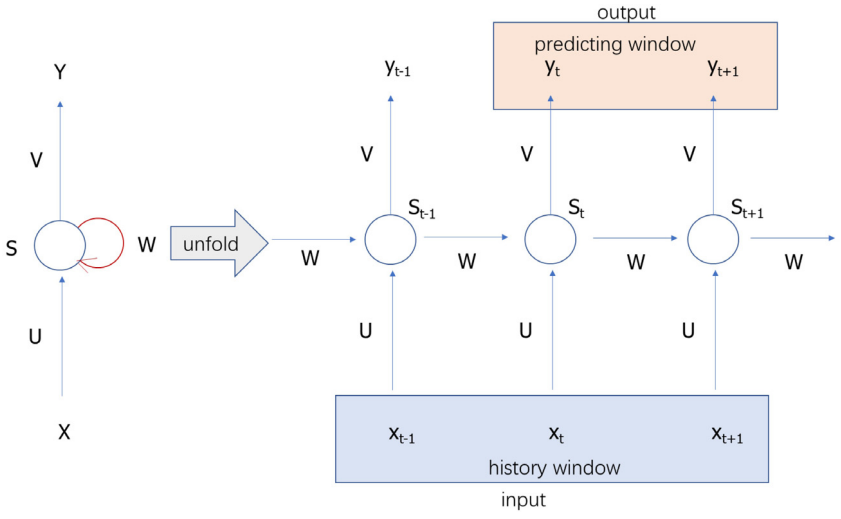


Fig. 2. RNN structure.

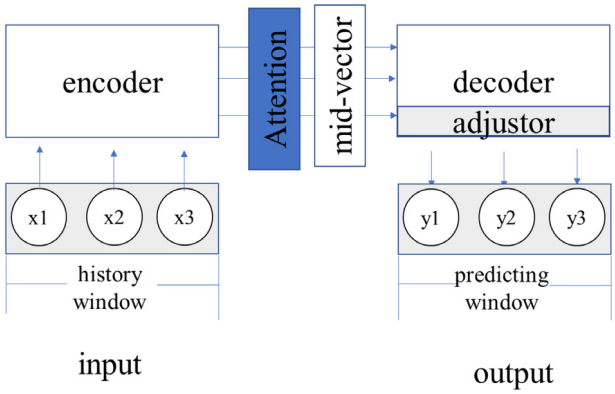


Fig. 3. Attention mechanism.

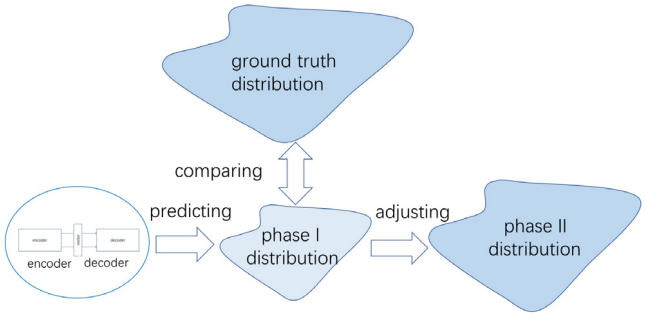


Fig. 4. Adjustor framework.

in Fig. 2, Fig. 3, and Eq. (1). In theory, recurrent neural networks can process sequences of arbitrary length. However, in practice, if the history window or the predicting window is too long or too short, it will lead to the vanishing gradient problem [35]. In general, the more history information, the more helpful it is for future predictions. However, we found that if the sequence of the historical logs is too long, that is, the history window value is too large, it will lead to a decline in the prediction effect. We suspect that the reason is that the recent sequence has too much impact on the network. Therefore, we need to change the size of the window according to the actual situation.

3.5. Adjustment factor definition

Our method can predict the logs of bigdata system. The types of logs could be errors, traffics, task volumes, warnings, etc. We found that the variance of the data predicted by the RNN network is often smaller than the variance of the true value. Therefore, in order to better fit the true data, we introduce an adjustor to adjust the output of the decoding module. As shown in Fig. 4, this adjustor spatially converts the distribution of the predicting data. We introduce a parameter F used to adjust the distribution of predictive data. F is calculated by

$$F = \frac{\mu_t}{\sigma_t} \times \frac{\sigma_p}{\mu_p}, \tag{8}$$

where μ_t is the location parameter of true data, σ_t is the scale parameter of true data, μ_p is the location parameter of predictive data and σ_p is the scale parameter of predictive data.

In phase I, we compute the location parameter and the scale parameter of the predictive data and the true data. Next we set a parameter called adjustment factor for our model. Then predictive data in phase I and the adjustment factor will be combined to obtain final predictive data in phase II by

$$f(x) = (x - \mu_p) \times exp(F) + \mu_p, \tag{9}$$

where x is the value in phase I and f(x) is the value in phase II.

3.6. Log pattern definition

The log pattern S is a finite set $\{D_1, D_2, \ldots, D_n\}$, where D_i (i = 1, …, n) is an attribute domain. The most common log pattern is {CaseID, message, @version, @timestamp, host}, which specifies the metadata describing the log, where CaseID, message, etc. are the attribute fields.

4. Our method

Our method is a comprehensive approach to smart computation and prediction of bigdata logs. Fig. 5 shows our method's framework. We will introduce the overall framework of our method (Section 4.1), implementation steps (Section 4.2), log fusion algorithm (Section 4.3), training and prediction model algorithm (Section 4.4) and diversified training set algorithm (Section 4.5).

4.1. Framework of our method

The data flow is from left to right, as shown in Fig. 5. First, we collect logs of individual components from the bigdata system. Second, the logs from these components are normalized into JSON format according to the rules we set. In this step, unstructured data is converted into structured data. Considering that massive logs require a distributed storage for persistence, the normalized logs are then stored in a distributed full text indexing system named Elasticsearch [36]. In this process we can normalize the heterogeneous logs. As a result, it will be very convenient to retrieve, analyze and process those logs. This also makes it easy for us to get the training set. Finally, when the data is ready, we can start to train the model and make predictions by using the trained model. In the next subsection, we will introduce our implementation steps of the training and predicting.

4.2. Implementation steps

As shown in Fig. 6, we design a pipeline of the training and predicting model. Next, we will describe the pipeline in detail when we introduce the specific implementation steps.

Step 1: Collect Logs

The bigdata log itself is a kind of bigdata, so we use the distributed ELK tools to collect, process, and query them. ELK is an acronym for the three open-source software respectively. They are Elasticsearch, Logstash [37], Kibana. Elasticsearch is an open source distributed search engine that provides three main functions for collecting, analyzing and storing. It is distributed, zero configuration, auto-discovery, automatic indexing, and has index copy mechanism, restful style interface, multiple data sources, automatic search load and so on. Logstash is mainly used for log collection, analysis, and filtering. It supports the combination with message queues to handle massive amounts of data without losing data. Based on the C/S architecture, whose agent should be installed on the nodes which need to collect logs. The server is responsible for filtering the logs from the nodes. Of course, we can use tools like kibana to analyze the logs with a friendly user interface.

We use Logstash to collect logs from various components of the Hadoop system (such as Hadoop, YARN, MapReduce, Hive, HBase, Spark, Solr, Carbondata, etc.). During the acquisition process, the log is formatted and normalized, such as Algorithm 1. Because

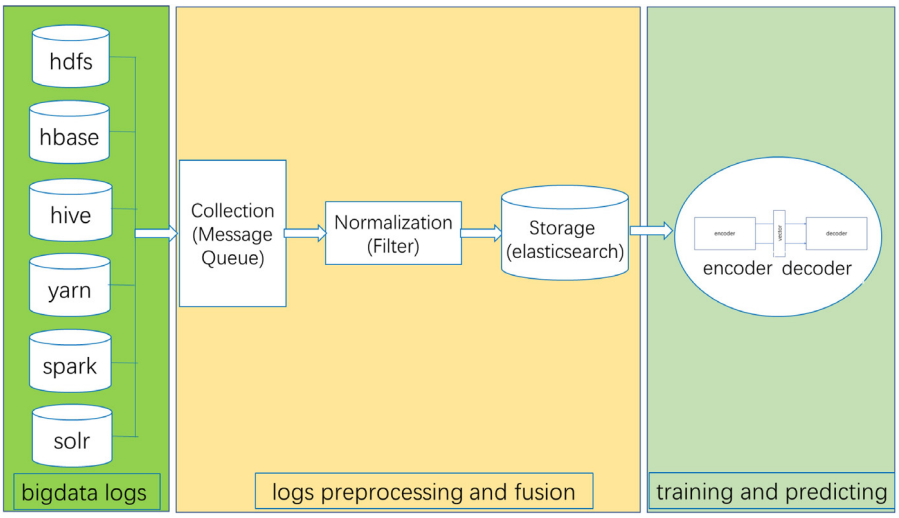


Fig. 5. The framework of our method.

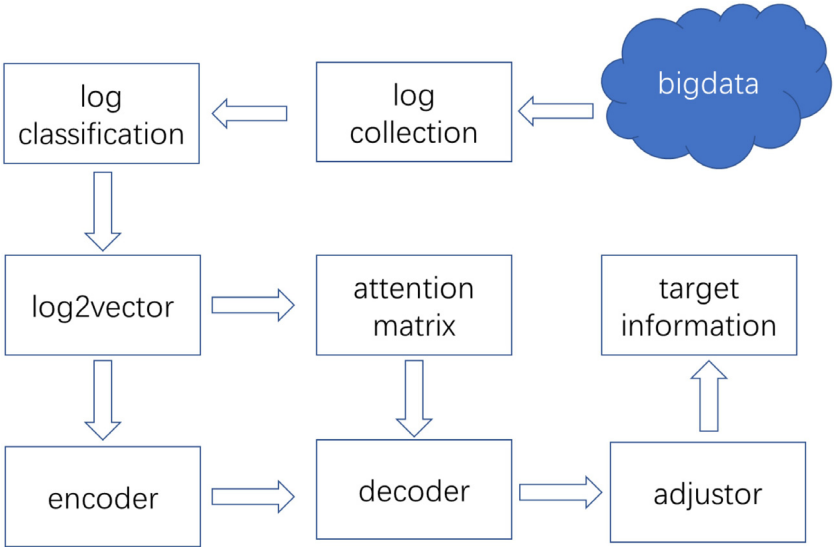


Fig. 6. The pipeline of training and predicting model.

the log information of an event is often entered into the log file in multiple lines, normalization is necessary. An example of a normalized log is given in Fig. 7. Therefore, we use the event as a granularity to normalize. Finally, the processed log data is stored in Elasticsearch, a parallel storage system with full-text search function. In this step, the collection and preliminary pre-processing of raw log data are completed, noise logs are eliminated, and multi-source heterogeneous logs are processed. In this way, we can easily get structured log information.

Step 2: Classify Logs

Since the data volume of the bigdata log is huge, we categorize the log events in order to simplify the calculation. And we use the category as the granularity for predictive analysis. We need to categorize the logs based on components, log levels (information, warnings, errors) and specific log key content (start, finish). Some algorithms like the cosine similarity, kmeans and pattern matching are used to generate a category dictionary after collecting logs for a period of time. We create a dictionary for mapping logs and categories in this step. If the input log does not exist in the dictionary, we mark it as $\langle UNK \rangle$. Once we obtain this dictionary, we can add a classification tag to each log before storing the logs in Elasticsearch in the previous step.

Step 3: Vectorize Logs

Each normalized, formatted, and categorized logs will have an ID (identified automatically in Elasticsearch). Those IDs are inserted into a set. Each element in this set is unique. Next, we build a dictionary based on this set. The key of this dictionary is used as the identifier for each log. Then the log is converted into one-hot vector, the vector of each log is trained by n-gram (the dimension of the vector is much smaller than the one-hot vector). Finally, the vector of each log is stored in a new dictionary. Therefore, those logs turn into vectors with dimensions. The vectors are going to be provided for the next two steps, one for the encoding layer as input data and the other for the attention matrix. We also call this step log2vector.

Step 4: Diversify training set

In many cases, we are not sure about the size of history window and the target window. In order to fit the size of different history window and predicting window, we need to make the training set more diversified without changing the original information. We set several hyperparameters, such as the maximum historical window, the maximum prediction window, and the number of training sets that need to be generated. Then, we generate a training set by using Algorithm 3. The training set imbalance problem is also an issue

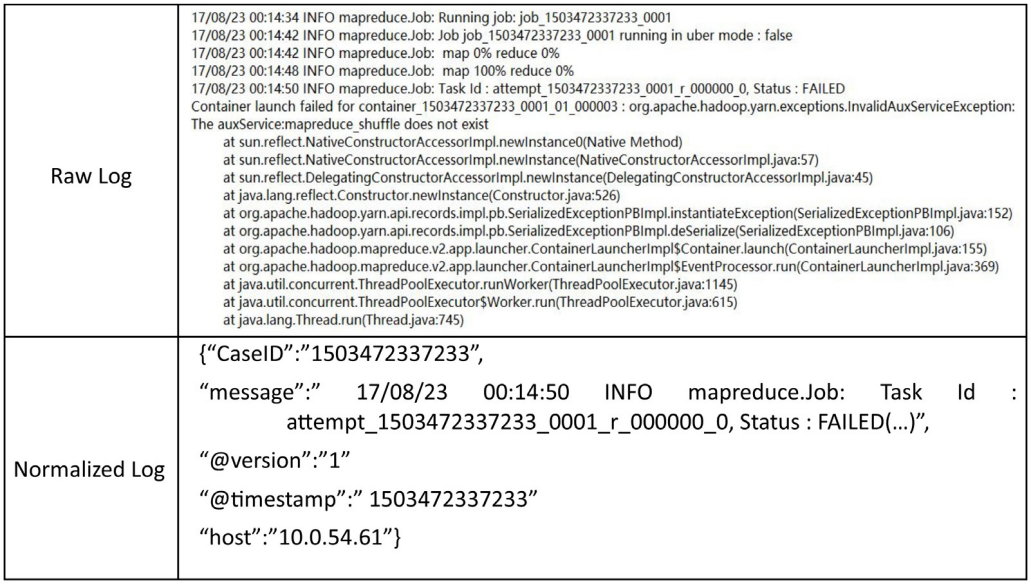


Fig. 7. An example of a normalized log.

to consider. We need to avoid low frequency occurrences of logs being annihilated by high frequency logs. One solution is to cluster fixed-length log sequences and then sample them with balanced number from every clusters.

Step 5: Encode Sequences

The input multi-source heterogeneous log data is vectorized into log2vectors of the same dimension in step 3. Next, we input it to the encoding layer in chronological order. We set a maximum history window size. Then the encoding layer will convert sequence into a hidden state vector. We use ⟨*PAD*⟩ notation for the log sequence smaller than the history window and use ⟨*UNK*⟩ notation for the unclassified log vector.

Step 6: Decode Vectors

The hidden layer state vector obtained from the previous step is multiplied by an attention matrix as the input to the decoding layer. And the target information is generated by the decoding layer. This target information is also a sequence. This sequence will artificially add a start symbol ⟨*START*⟩ and an end symbol ⟨*END*⟩. The decoded sequence also needs to be processed by an adjustor so that the distribution of the predicting data is better fit to the real data distribution.

Step 7: Get Target Information

The target information is the predicting sequence we need to obtain. The target information may be the system information (cpu utilization, memory utilization, network traffic) of each node in the bigdata system. It may come from some components, some application system, or security system. We should determine what kind of target information we need before the training step. Then we give the label of the training set (target information of real data). During training, our model will learn based on the input sequence and the actual output sequence. Once the model is generated after training, we can predict the target information we want in the predicting step.

4.3. Log fusion algorithm

In the process of collecting logs, it is necessary to convert the logs (unstructured text data) from each component of bigdata according to certain rules. We remove noise data, add timestamps, add classification labels. After this process, these log files become structured data. See Algorithm 1.

Algorithm 1: Log Fusion Algorithm

Input: The set of log path, *LogPath*; The path of ElasticSearch, *ESPath*; The name of index for ElasticSearch, *IndexName*; The name of table for ElasticSearch, *TableName*; The configure of log filter, *FilterConfig*; The rule of regular expression, *RuleRegex*;

Output: Standard event based log, *StandardLog*;

1: Extracting the set of raw log from *LogPath* into a data stream;
2: Processing log data stream according to *FilterConfig* and *RuleRegex* to obtain event-based and structured logs;
3: Appending host IP, timestamp, version number, etc. to the log;
4: Saving standard logs into *ESPath* by specified *IndexName* and *TableName*;
5: **return** *StandardLog*;

In the log normalization process, the first task is to distinguish between the event boundaries in the log text stream because the logs output by various components of the bigdata system often have their own format. In many cases, many rows are output for an event log. As shown in Fig. 7, when a mapreduce task fails, the log will output multiple rows in the form of a stack to represent the error message. Then we set some rules by regular expression to normalize the logs based on the boundary of the event. The study of time series data necessarily makes it very important to record timestamps (logs with event granularity) because we need them to record the order. Note that we have to customize different filter configuration files for different bigdata components.

4.4. Training and predicting algorithms

Our algorithm is mainly based on the seq2seq of the recurrent neural network. This algorithm consists of two options, one is training and the other is predicting. In training, the bigdata logs changed into vectors are input into the neural network to train a predictive model. In this phase, the neural network model can obtain a large number of weights and bias. It contains the context information and relationships among the logs. Thus it has the ability to predict a future log sequence if given a log sequence.

In predicting, we can input the real-time generated log sequence into the trained model to obtain the predictive target data. The Algorithm 2 describes our specific approach.

Algorithm 2: Training And Predicting Algorithms

Input: The data *StandardLog* stored in Elasticsearch is the input data for neural network, Optional parameter *Flag* is used to select training or prediction, Weight matrix *AttentionWeight* generated by the probability of the log appearing, Log history window *HistoryWindow*, Log target window *TargetWindow*

Output: Trained model *TrainedModel*,Target Log Data *TargetData*

```
SourceLogs = Open(StandardLog);
Vocab = ExtractLogVocab(SourceLogs);
Embedding = GetEmbedSequence(Vocab);
EmbeddingInput = GetInput(Embedding, SourceLogs, HistoryWindow);
MiddleVector = Encoder(EmbeddingInput);
DecoderEmbedInput = ProcessDecoderInput(MiddleVector , AttentionWeight, Flag);
if Flag ≡ training then
    TrainingHelper = GetTrainHelper(DecoderEmbedInput, TargetWindow);
    TrainingDecoder = GetBasicDecoder(LstmCell, TrainingHelper);
    TrainingDecoderOutput = GetDynamicDecode(TrainingDecoder);
    for epochs do
        Loss = CrossEntropy(TrainingDecoderOutput, label);
    return TrainedModel;
if Flag ≡ predicting then
    PredictingHelper = GetPredictHelper(DecoderEmbedInput, start, end);
    PredictingDecoder = GetBasicDecoder(LstemCell, PredictingHelper);
    PredictingDecoderOutput = GetDynamicDecode(PredictingDecoder);
    if TrainedModel is exist then
        loader.restore(TrainedModel);
        TargetData = sess.run(StandardLog);
        return TargetData;
final ;
```

4.5. Diversifying training set

In previous seq2seq model, an encoder neural network reads and encodes a source sentence into a fixed-length vector. However, our goal is to make our model more flexible, without fixing the size of the historical window and the forecast window. Therefore, we need to generate more training data for this goal. We specify two hyperparameters, which are the maximum history window size and the maximum prediction window size. Then, a random-sized original sequence is generated at a random position from the real data. And a prediction sequence that does not exceed the size of the prediction window is generated starting from the next token of this sequence. To better illustrate this process, we show the algorithm in Algorithm 3.

We randomly obtain different sizes of history and predict sequences from the original log sequence through the above algorithm. These sequences provide the model with rich resources for learning. Once training is complete, the model can accept sequences of different lengths and output different length prediction sequences. In our experience, when the number of diversified training set is higher than the number of vocabulary by more than 100 times, accuracy often reaches almost 80% of the optimal value.

Algorithm 3: Algorithms of Diversifying Training Set

Input: The data *StandardLog* stored in Elasticsearch is the input data for neural network, Maximum Log history window $MaxHistoryWindow$, Maximum Log target window $MaxTargetWindow$, Total number of Training Set *Total*

Output: Training Set *TrainingSet*

```
for Total do
    HistoryInt = Random(1,MaxHistoryWindow);
    TargetInt = Random(1,MaxTargetWindow);
    Index=Random(StandardLog);
    TrainingSet.append(StandardLog[(Index - HistoryInt), (Index + TargetInt)])
return TrainedModel;
final ;
```

5. Experiment

5.1. Experimental data and environment

Our experimental data is a set of log files of various components of Hadoop's bigdata system provided by Beyondsoft (Shanghai) Co., Ltd. This system consists of 32 nodes that were installed HDFS, HBase, Hive, Spark, Solr, YARN and other components. We collected one-year log data in 2017 into Elasticsearch. Then we trained the data on a machine equipped with Nvidia GTX1080 GPU graphics. The total size of raw log data that derived from hadoop family components is around 1.08TB.

5.2. Experimental tasks and evaluation indicators

This article uses several indicators to evaluate the proposed algorithm model. These indicators consist of root mean square error (RMSE), extended variance score (ESV), mean absolute error (MEANAE), mean absolute error (MEDIANAE), R2 SCORE, the standard deviation of the prediction data and the standard deviation of the real data.

The equation for root mean square error is

$$RMSE(\bar{Y}, Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{Y} - Y)^2}, \tag{10}$$

where Y is the true value, $\bar{Y}$ is the predicted value, and n is the number of data items. In general, the smaller the RMSE error values, the better the accuracy of the model.

Explained variation measures the proportion to which a mathematical model accounts for the variation (dispersion) of a given data set. The extended variance score is calculated by

$$EVS(\bar{Y}, Y) = 1 - \frac{Var(\bar{Y} - Y)}{Var(\bar{Y})}, \tag{11}$$

where Y is an array of real values and $\bar{Y}$ is an array of predicted values. The closer this value is to 1, the better the fit between these two data.

Mean absolute error is mainly used to measure the average difference between the predicted value and the true value. It is calculated by

$$MEANAE(\bar{Y}, Y) = \frac{1}{n}\sum_{i=1}^{n}|\bar{y}_i - y_i|, \tag{12}$$

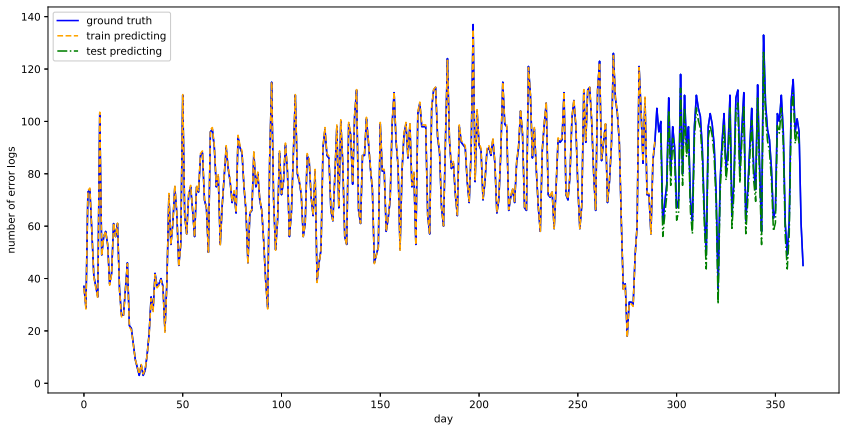


Fig. 8. Comparison of predicted and true values (365 days).

where y_i is the true value, $\bar{y}_i$ is the predicted value, and n is the number of data items.

$$MEDIANAE(\bar{Y}, Y) = median(|\bar{y}_1 - y_1|, \ldots, |\bar{y}_n - y_n|) \tag{13}$$

Eq. (13), which is suitable for measuring datasets that contain outliers, is median absolute error.

$$R^2(\bar{Y}, Y) = 1 - \frac{\sum_{i=1}^{n}(\bar{y}_i - y_i)}{\sum_{i=1}^{n}(\bar{y}_i - mean(Y))} \tag{14}$$

Eq. (14) shows how much the dependent variable is affected by the change in the independent variable, or how well the regression equation fits the observed value.

5.3. Results

We trained and predicted error logs from "Hyren Bigdata Platform" that manages the hadoop bigdata system. The log data was selected 80% as the training set, 20% as the test set. The history window is set to 10 and the predicting window is set to 1. As shown in Fig. 8, the main result shows that the log data predicted by our proposed method has a very good fit for the training dataset (R2 approx. 1) and the test dataset (R2 approx. 0.94).

We take the prediction results of the 20-day test dataset in order to check the effect of the fitting intuitively. Fig. 9(b) line does not use the adjustor, Fig. 9(c) line adds the adjustor and Fig. 9(a) line uses random forest [38] algorithm. Those figures show that the algorithm presented in this article has better predictions with the addition of the adjustor. Furthermore, compared with the machine learning algorithm, it can be found that the proposed method generates a prediction curve closer to the true value curve.

We also tried several traditional algorithms for comparison. These algorithms include Random Forest Model, AdaBoost Model, Bagging Model, Gradient Boosting Model, ARIMA and HMM. All the evaluation results are shown in Table 1 and bold characters represent the optimal value. By using the algorithm of this paper, the R2 value of the predictive data and the training set was increased from 0.88 to approximately 1, and the R2 value of the predictive data and the test set was increased from 0.82 to 0.94. The best R2 value of the previous methods is 0.7 in training set and 0.35 in test set. And those RMSE values are generally above 10. However, our algorithm can achieve RMSE value below 5. Our proposed algorithm gets a score of R2 closer to 1, representing a very good fit between the predicted value and the true value. Compared with the previous predictive method, our method's RMSE value is reduced by 46.65% and R2 value fitting degree is improved by 14.28%. It shows that the proposed method can achieve very good predictive analysis results for massive, multi-source heterogeneous logs. Obviously, the method proposed in this paper outperforms all the previous methods.

To better illustrate the effectiveness of our approach, we compared log analysis results for other components of bigdata. Table 2 shows that each indicator reflects a good fit between the prediction data and the true data. It shows that the proposed algorithm has good scalability.

5.4. Discussions

Because traditional methods and machine learning only have a short-term memory, they are not suitable for massive data prediction. Although RNN has limited short-term memory, it suffers from severe gradient disappearance problems during the use of backpropagation and gradient descent algorithms. A pure LSTM network cannot introduce attention mechanisms to give more weight to a particular log. The Seq2seq network can use the log that has just passed in conjunction with the current log as input to predict which logs will occur next. This brings the advantage of limited short-term memory. When seq2seq have enough background information, it can predict the next most likely log. The large weight matrix of the Seq2seq network is appropriate for fitting massive time series data from bigdata system.

However, we also find that our method of predicting logs is not always effective. The prediction of a random log sequence without any regularity will be unsatisfactory. In some experiments, we found that our method can have a good performance on the logs with some characteristics, such as periodicity, trend or holiday effects. Since our method selects the most likely log sequence from the vocabulary, it is necessary to have the log in the vocabulary. Our method cannot predict the logs that have never appeared. For the log that is not in the vocabulary, a similarity algorithm can be used to find the most similar log in the vocabulary for replacement.

6. Conclusion

In this paper, based on the massive, multi-source and heterogeneous characteristics of the log of many components of bigdata platform, a method of predictive analysis is proposed. When people want to be able to predict future events of the IoT system based on massive logs, the method we propose can accomplish this task very well. This method is based on a seq2seq recurrent neural network model, combining a distributed log collection system, log normalization, event-based and vectorization processing, encoding algorithms, attention mechanisms, adjustors and decoding

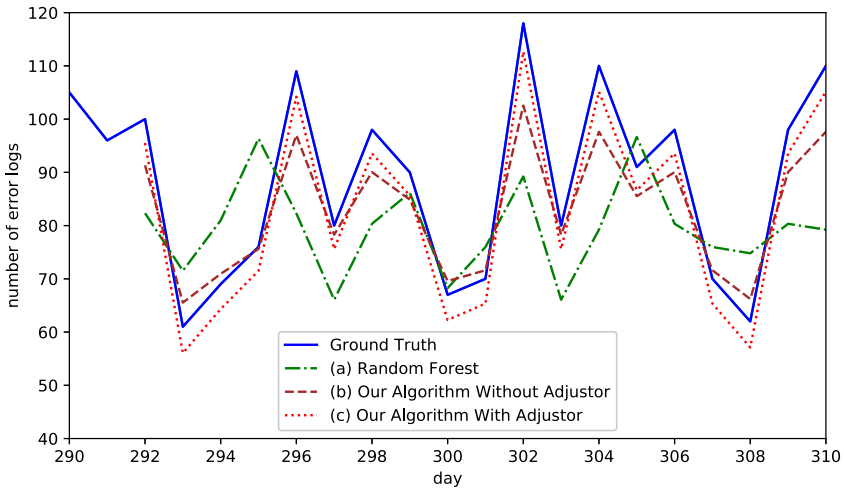


Fig. 9. Comparison of predicted and real values (20 days).

Table 1
Comparison of various algorithms.

Dataset	Algorithm name	RMSE	EVS	MEAN	MEDIA	R2	δ_p	δ_t
train	Our Method(no adjustor)	8.68	0.88	6.8	5.5	0.88	16.85	25.55
	Our Method($F = 1.53$)	**0.56**	**1**	**0.5**	**0.55**	**1**	**25.78**	
	Original LSTM	18.1	0.5	14.05	11.13	0.5	17.58	
	Random Forest Regression	14.05	0.7	10.62	7.16	0.7	20.78	
	AdaBoot Regressor	12.25	0.77	9.35	7.58	0.77	18.58	
	Bagging Decision Tree	14.23	0.69	10.57	8.72	0.69	20.81	
	Gradient Boosting	12.44	0.76	9.01	7.2	0.76	19.67	
	ARIMA(7, 1, 0)	17.19	0.55	12.82	9.9	0.55	21.31	
test	Our Method(no adjustor)	8.21	0.88	6.88	6.71	0.82	12.75	20.11
	Our Method($F = 1.53$)	**4.7**	**1**	**4.68**	**4.56**	**0.94**	**19.51**	
	Original LSTM	21.66	−0.13	17.4	15.08	−0.21	10.97	
	Random Forest Regression	15.79	0.49	12.84	10.37	0.35	12.58	
	AdaBoot Regressor	12.23	0.63	10.6	8.9	0.54	10.52	
	Bagging Decision Tree	16.21	0.46	12.29	9.49	0.32	13.67	
	Gradient Boosting	15	0.56	11.77	9.88	0.41	11.19	
	ARIMA(7, 1, 0)	17.48	0.24	14.14	12.43	0.24	0.24	
	HMM	26.33	−0.23	22.24	19.52	−0.24	23.6	

Table 2
Predictive analysis of our algorithm in other logs.

Dataset	Log name	RMSE	EVS	MEAN	MEDIA	R2	δ_p	δ_t
train	HDFS error log(no adjustor)	5.12	0.84	4.34	4.06	0.84	7.82	15.88
	HDFS error log($F = 2.03$)	1.07	0.99	0.85	0.71	0.99	7.82	15.88
	A host acess Hbase(no adjustor)	4.97	0.87	4.09	3.66	0.86	8.47	13.36
	A host acess Hbase($F = 1.03$)	0.86	1	0.73	0.56	1	12.96	13.36
	B job call in Yarn(no adjustor)	22.8	0.84	15.18	11.88	0.84	33.35	56.04
	B job call in Yarn($F = 1.09$)	5.22	0.99	3.61	2.81	0.99	51.02	56.04
test	HDFS error log(no adjustor)	10.05	0.84	3.93	2.16	0.84	14.68	15.88
	HDFS error log($F = 2.03$)	2.23	0.99	0.88	0.51	0.99	22.46	15.88
	A host acess Hbase(no adjustor)	9.79	0.85	4.3	2.9	0.85	15.33	24.83
	A host acess Hbase($F = 1.03$)	2.28	1	1.69	1.62	0.99	23.46	24.83
	B job call in Yarn(no adjustor)	11.94	0.83	9.98	9.01	0.82	16.57	29.29
	B job call in Yarn($F = 1.09$)	4.26	0.99	3.79	4.03	0.98	25.35	29.29

algorithms and other module processes. The model obtained after learning the log has a very good predictive analysis effect. After comparing with several traditional predictive methods, we found there were improvements in RMSE, R2, and other performance indicators. The RMSE has dropped by at least 46.65%, and the R2 score has increased by at least 14.28%. This shows that the method proposed in this paper has better prediction accuracy. Today is the era of bigdata and artificial intelligence. By combining the algorithms from bigdata and artificial intelligence, this paper focuses on bigdata log predictive analysis to better analyze the past, present and future of bigdata's status. We expect to improve smart computing of cognitive Internet of Things.

The accuracy of our method may be improved in two aspects. One is to improve the attention mechanism to get a more accurate sequence. The other is to introduce the knowledge graph because the relationship among logs is important information. If the neural

network can get this information, it is expected to become more intelligent.

Acknowledgments

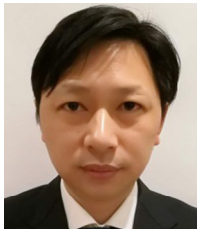
This work of this paper is supported by the Shanghai Innovation Action Plan Project under Grant 16511101200. The work of this paper is also supported by National Natural Science Foundation of China under Grant No. 61728202-Research on Internet of Things Big Data Transmission and Processing Architecture based on Cloud-Fog Hybrid Computing Model, China & Grant No. 61572137-Multiple Clouds based CDN as a Service Key Technology Research, China, and Shanghai 2016 Innovation Action Project, China under Grant 16DZ1100200-Data-trade-supporting Big Data Testbed. Shanghai 2018 Innovation Action Plan - Hong Kong, Macao and Taiwan Science and Technology Cooperation Project under Grant No. 18510760200-Research on Smart City Big Data Processing Technology based on Cloud-fog Mixed Mode.

References

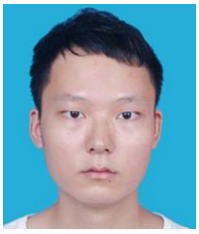
[1] Andrew Meloa, The US government is pouring money into the Internet of Things, https://www.businessinsider.com.
[2] Mauro Coccoli, Paolo Maresca, Lidia Stanganelli, The role of big data and cognitive computing in the learning process, J. Visual Lang. Comput. 38 (2017) 97–103.
[3] Fares A. Nassar, Dorothy M. Andrews, A methodology for analysis of failure prediction data, in: IEEE Real-Time Systems Symposium, 1985, pp. 160–166.
[4] Shilin He, Jieming Zhu, Pinjia He, Michael R. Lyu, Experience report: System log analysis for anomaly detection, in: IEEE International Symposium on Software Reliability Engineering, 2016, pp. 207–218.
[5] Adam Oliner, Archana Ganapathi, Wei Xu, Advances and challenges in log analysis, Commun. ACM 55 (2) (2012) 55–61.
[6] T.T.Y. Lin, D.P. Siewiorek, Error log analysis: statistical modeling and heuristic trend analysis, IEEE Trans. Reliab. 39 (4) (2002) 419–432.
[7] Renta Iváncsy, István Vajk, Frequent pattern mining in web log data, Acta Polytech. Hung. 3 (1) (2006) 223–228.
[8] Robiah Yusof, Siti Rahayu Selamat, Shahrin Sahib, Intrusion alert correlation technique analysis for heterogeneous log, Int. J. Comput. Sci. Netw. Secur. 59 (9) (2008) 132–138.
[9] H. Asifiqbal, Nur Izura Udzir, Ramlan Mahmod, Abdul Azim Abd. Ghani, Filtering events using clustering in heterogeneous security logs, Inf. Technol. J. 10 (4) (2011).
[10] Pinjia He, Jieming Zhu, Shilin He, Jian Li, Michael R. Lyu, Towards automated log parsing for large-scale log data analysis, IEEE Trans. Dependable Secure Comput. PP (99) (2017) 1–1.
[11] Dionysios Logothetis, Chris Trezzo, Kevin C. Webb, Kenneth Yocum, In-situ MapReduce for log processing, in: Usenix Conference on Hot Topics in Cloud Computing, 2012, pp. 26–26.
[12] Hongyong Yu, Deshuai Wang, Mass log data processing and mining based on Hadoop and cloud computing, in: International Conference on Computer Science & Education, 2012, pp. 197–202.
[13] Hemant Hingave, Rasika Ingle, An approach for MapReduce based log analysis using Hadoop, in: International Conference on Electronics and Communication Systems, 2015, pp. 1264–1268.
[14] Sandeep Kumar Dewangan, Shikha Pandey, Toran Verma, A distributed framework for event log analysis using MapReduce, in: International Conference on Advanced Communication Control and Computing Technologies, 2017, pp. 503–506.
[15] R.A. Epperly, G.E. Heberlein, L.G. Eads, A tool for reliability and safety: Predict and prevent equipment failures with thermography, 49(11) (1997) 59–68.
[16] Rich Caruana, Alexandru Niculescu-Mizil, An empirical comparison of supervised learning algorithms, in: International Conference on Machine Learning, 2006, pp. 161–168.
[17] Rich Caruana, Nikos Karampatziakis, Ainur Yessenalina, An empirical evaluation of supervised learning in high dimensions, in: International Conference on Machine Learning, 2008, pp. 96–103.
[18] Christopher Brooks, Craig Thompson, Stephanie Teasley, A time series interaction analysis method for building predictive models of learners using log data, in: International Conference, 2015, pp. 126–135.
[19] Lawrence R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Read. Speech Recognit. 77 (2) (1990) 267–296.
[20] George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Time series analysis: Forecasting and control (Revised Edition), J. Market. Res. 14 (2) (1994) 199–201.
[21] Y. Romahi, Q. Shen, Dynamic financial forecasting with automatically induced fuzzy associations, in: IEEE International Conference on Fuzzy Systems, 2000. Fuzz IEEE, vol. 1, 2000, pp. 493–498.
[22] Saratha Sathasivam, Wan Ahmad Tajuddin Wan Abdullah, Logic learning in hopfield networks, Modern Appl. Sci. 2 (3) (2008) 57.
[23] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[24] K Greff, R.K. Srivastava, J Koutnik, B.R. Steunebrink, J Schmidhuber, LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2017) 2222–2232.
[25] Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever, An empirical exploration of recurrent network architectures, in: International Conference on International Conference on Machine Learning, 2015, pp. 2342–2350.
[26] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, Comput. Sci. (2014).
[27] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to sequence learning with neural networks, 4 (2014) 3104–3112.
[28] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural machine translation by jointly learning to align and translate, Comput. Sci. (2014).
[29] Allen Newell, Herbert A. Simon, Human Problem Solving, Prentice-Hall, 1972.
[30] Dharmendra S. Modha, Rajagopal Ananthanarayanan, Steven K. Esser, Anthony Ndirango, Anthony J. Sherbondy, Raghavendra Singh, Cognitive computing, Commun. ACM 54 (8) (2011) 62–71.
[31] Amit Sheth, Internet of things to smart IoT through semantic, cognitive, and perceptual computing, IEEE Intell. Syst. 31 (2) (2016) 108–112.
[32] Alex Graves, Generating sequences with recurrent neural networks, Comput. Sci. (2013).
[33] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, Yoshua Bengio, Attention-Based models for speech recognition, Comput. Sci. 10 (4) (2015) 429–439.
[34] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, Show, attend and tell: Neural image caption generation with visual attention, Comput. Sci. (2015) 2048–2057.
[35] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training recurrent neural networks, 52 (3) (2012) III–1310.
[36] Oleksii Kononenko, Olga Baysal, Reid Holmes, Michael W. Godfrey, Mining Modern Repositories with Elasticsearch, ACM, 2014, pp. 328–331.
[37] Sushma Sanjappa, Muzameel Ahmed, Analysis of logs by using logstash, 2017.
[38] L. Breiman, Random forest, Mach. Learn. 45 (2001) 5–32.

Pin Wu is an Associate Professor, Shanghai University. She received the B.S. and Ph.D. at Nanjing University of Science and Technology. She had worked in Zhejiang University as a post-doctor for two years, and had worked in Michigan State University as a senior visiting scholar for one year. Her current research interests are focusing on high performance computing (HPC), computational fluid dynamics (CFD) and image processing. Over the past ten years, she has published over 30 technical papers in the related fields. She will keep on the research work addressing Cross-disciplinary of computer and mechanics.

Zhihui Lu is an Associate Professor in School of Computer Science, Fudan University. He received a Ph.D. computer science degree from Fudan University in 2004, and he is a member of the IEEE and China computer federation's service computing specialized committee. His research interests are big data architecture, cloud computing and service computing technology, edge computing, smart city and IoT.



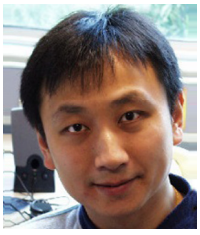
Quan Zhou is a graduate student at Shanghai University, master's degree at Fudan University. His research interests are mainly big data, machine learning, neural networks, knowledge graphs, and data analysis. Has many years of experience in the construction and operation of software systems and network systems in the financial industry (securities, banking and insurance) and manufacturing.



Zhidan Lei is a graduate student at Shanghai University. His research fields are mainly machine learning, neural networks, NLP and other fields.

Xiaoqiang Li is an Associate Professor at Shanghai University, Ph.D. at Fudan University. His major research areas are image processing and pattern recognition, machine learning, video content understanding and analysis, and computer vision. He is currently an IEEE member, ACM member, a senior member of the Chinese Computer Society, and a deputy director of the Multimedia Special Committee of the Shanghai Computer Society.

Meikang Qiu received the B.E. and M.E. degrees from Shanghai Jiao Tong University and received Ph.D. degree of Computer Science from University of Texas at Dallas. Currently, he is an Adjunct Professor at Columbia University and Associate Professor of Computer Science at Pace University. He is an IEEE Senior member and ACM Senior member. He is the Chair of IEEE Smart Computing Technical Committee. His research interests include cyber security, cloud computing, big data storage, hybrid memory, heterogeneous systems, embedded systems, operating systems, optimization, intelligent systems, sensor networks, etc. He has published 5 books, 330 peer-reviewed journal and conference papers (including 150+ journal articles, 180+ conference papers, 50+ IEEE/ACM Transactions papers), and 3 patents.



Patrick C.K. Hung is a Professor at the Faculty of Business and Information Technology in University of Ontario Institute of Technology. Patrick has been working with Boeing Research and Technology in Seattle, Washington on aviation services-related research projects. He has published more than 100 journal and conference papers on the fields of services computing, cloud computing, big data, business process and security.