A Mining Algorithm of Frequent Items in Data Streams Based on Apache Storm

Weihua Hu[1, a], Ziang Guo[2, b] and Mingzhong Chen[2, c]

[1]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China;

[2]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China;

[3]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China;

[a] hwh@hdu.edu.cn, [b]iziang@yeah.net, [c]cmzalvin@163.com

Keywords: Data dreams, Data mining (DM), Frequency items, ε －approximation.

Abstract. Frequent items in data streams refer to log items appeared in numerous data streams that are over specified thresholds. Under data stream model, data streams are continuous, and the algorithm can only scan data once. Furthermore, data streams are unlimited, while the available storage space is limited. Therefore, it is usually not possible to excavate all accurate frequent items in data. The paper put forward the improved algorithm WFIC of digging approximate frequent items in data streams, and the algorithm can dynamically adjust weight of all kinds of data, which will make the rank of significant data more top-ranked. Experimental results show that the algorithm has good performance, which can effectively determine frequent items in data streams.

Introduction

For large data, generally there are two modes of batch computation and stream-oriented computation. Batch computation fits for scenes of low requirements for instantaneity and high requirements for data accuracy and reliability. However, stream-oriented computation fits for scenes of strict requirements for instantaneity, while low requirements for data accuracy[1].

For large Internet enterprises, logs produced each day is huge, and one of the basic needs of enterprises is finding out frequent items in numerous logs[2]. This paper has come up with a mining algorithm of frequent items in Data Streams based on Apache Storm processing system, by classifying the input data stream and dynamically endowing different weights, the mining frequent items of data stream can reflect the actual operating situation of the system.

Introduction of related work

Introduction of the basic concept

First, we will define some concepts of approximation algorithm of mining frequent items in data stream:

Imagine that $s \in (0,1)$ and $\varepsilon \in (0,1)$ are respectively the specified support and error by users, and N is the current data number of data stream that have arrived. At any moment, when users take frequent items query, if the algorithm can guarantee the output result will satisfy:

(1) All data items with actual frequency over sN are output;

(2) All data items with actual frequency under $(s-\varepsilon)N$ won't output;

And if error between frequency value estimated by the algorithm and actual frequency is less than ε N, then it is considered that the output result of the algorithm satisfies approximate requirements of ε - .

Introduction of related studies

At present, mining algorithm of frequent items in data stream can basically be divided into Hash and sampling, and they will be separately introduced in the following.

The basic idea of the method based on Hash is to map range of data stream to the Hash table in memory, and each bucket of Hash table corresponds a counter. Based on Hash method, Charikar and others provide the method of Count Sketch[3], which needs storage space of $O(k/\varepsilon^2 \log N/\delta)$, and it takes the probability of $1\text{-}\delta$ to output all data items that have appeared over $1/(k+1)$; Jin and others have provided hCount algorithm[4], and the algorithm will take space of $O(\varepsilon^{-1}\log(-M/\log\delta)$.

Sampling is another common technology to mine frequent items in data stream. Manku and Motwani have provided a certain approximation algorithm of ε-—Lossy Counting algorithm[5]. The space complexity of Lossy Counting algorithm is $\mathrm{O}(1/\varepsilon \log \varepsilon N)$. Demaine and other people take k counters, and output frequent items of which the occurrence frequency is over 1/(k+1). However, under the situation that data distribution in data stream is unknown, their algorithm cannot determine the error scope of the output frequency values[6].

WANG Wei-Ping and others have provided with the elgorithm of EC[7]. However, this algorithm has not distinguished types of input and output, which are really important in the mining of frequent items in data stream. The paper has provided with WFIC algorithm on the basis of EC algorithm, and this algorithm has classified data stream with different weights. Moreover, it has implemented the algorithm on Apache Storm processing system, which achieves good effects.

Algorithm description and Storm topology design

Algorithm description

WFIC algorithm requires that users should firstly specify error ε, while the support s has been given when users issue the query. s can be any value in (ε,1). Algorithm will fixedly reserve $1/\varepsilon$ samples in data stream, and dynamically maintain sample set with the continuous arrival of data in data stream. At any moment, when user query support is over the frequent item of s, WFIC algorithm will take advantage of sample set to give an approximate result. In the following, detailed definition of WFIC algorithm will be given.

WFIC algorithm takes sample set D to reserve $1/\varepsilon$ samples, and each sample is 5 tuple $<e, w_e, f, N_e, df>$, among them e is one item in data stream; w_e is the corresponding weight of log item e, and w_e is positive integer. f and df are two counters, Ne is the number of data item comes from data stream when the 5 tuple has added to sample set D. At any moment, take N to indicate the arrival number of data items in the current data stream. Imagine that the new arrival tuple is e, the execution step of WFIC algorithm is shown in the following:

Step 1: the value of N plus 1. If e is in D, perform step 2, otherwise perform step 3.

Step 2: update counter f corresponding data item e in set D, and make $f = f + w_e$, then return to step 1 at the arrival of new log item.

Step 3: estimate whether D is full. If not, perform step 4, otherwise perform step 5.

Step 4: add 5 tuple $<e, w_e, N, 0, w_e>$ to set D, then return to step 1 at the arrival of new log item.

Step 5: if there is no 5 tuple of f=0 in D, then perform step 6, otherwise perform step 7.

Step 6: traverse set D, reduce k from each f in 5 tuple, and k is the minimum value of log item weight in the current sample set D, df plus 1. Then return to step 5.

Step 7: delete all 5 tuples that f=0 in D, and add new 5 tuple $<e, w_e, N, 0, w_e>$.

At any moment, when user query support is over the frequent item of s, the algorithm will scan sample set D, and all 5 tuples that satisfy $f + df > (s-\varepsilon)N$ are to be called frequent items with the support of over s.

Correctness analysis of algorithm

For data item 3, make η to indicate the actual frequency of current e in data stream, and make $S[i, j]$ to indicate the data set composed of all data items between i and j in data stream S.

In the following, tput results of WFIC algorithm can be verified to satisfy approximation requirements of ε - .

(1) All data items with actual frequency over sN are output;

Because all data items with actual frequency over ε N are in D, then all data items with actual frequency over sN must be in d. For each data item with actual frequency over sN, there exists $f + df > \eta - \varepsilon N$, so that η - εN > (s - ε)N. And for any 5 tuple $< e, w_e, f, N_e, df >$, there exists $f + df > \eta - \varepsilon N$, so that $f + df > (s - \varepsilon)N$, and the query algorithm will output all data items satisfied to $f + df > (s - \varepsilon)N$. Thus, data items with actual frequency over sN are output.

(2) All data items with actual frequency under $(s - \varepsilon)N$ won't output;

For each data item e in D that the actual frequency is lower than $(s - \varepsilon)N$, that is $\eta < (s - \varepsilon)N$. Because for each 5 tuple $< e, w_e, f, N_e, df >$, there exists $f + df <= \eta$, and $f + df < (s - \varepsilon)N$. From the algorithm, it can be known that such date items won't be putput.

(3) Error between frequency value estimated by the algorithm and actual frequency is less than ε N;

Because for each 5 tuple $< e, w_e, f, N_e, df >$, which simultaneously satisfies $f + df <= \eta$ and $f + df > \eta - \varepsilon N$, and it can be concluded that $\eta - \varepsilon N < f + df <= \eta$, the $f + df$ is the frequency value estimated by algorithm.

In conclusion, the output results of WFIC algorithm satisfy the approximation requirements of ε - .

Storm topology design

Storm is a streamed computing framework of Twitter open-source, we can take advantage of the Storm to make real-time calculation of streamed data. Similar to tasks in Hadoop, in Storm, we call each task topology, and this topology is a directed acyclic graph that defines the data direction, including where are data come from, they will go where, and what processing procedures are needed[8].

In this paper, we will take advantage of Apache Kafka message queue as the source of data stream. Imagine all data on the server are regarded as messages to be sent to Kafka cluster, then Storm will read data from Kafka to make analysis. We have classified logs, and the class information is taken in each log item.

In order to realize dynamic analysis of log weight, we have set another spout, and this spout will synchronize meta-information of all kinds of log weights from table MySQL regularly. In this way, Storm topolofy can be made continuously to dynamically adjust weight of all kinds of logs. The results of Storm topology results are shown in chart 1:

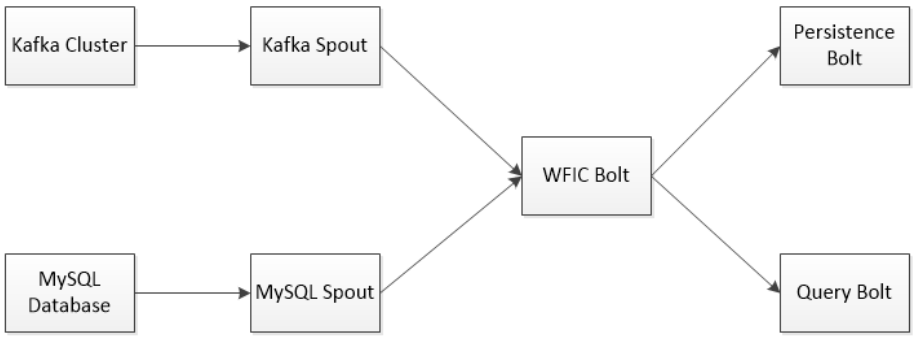

Chart 1 Topological graph of data stream

Experiment and analysis

We take logs produced by the real server, and test WFIC algorithm. The experiment has adopted machine A as the machine to produce logs, and another machine B will arrange Kafka cluster. Logs produced from machine A will be transmitted to machine B in real time, and other 5 machines will

constitute Strom cluster to read logs and make analysis from Kafka.

The experiment has adopted three types of logs produced by machine A, and they are separately Message log, Dmesg log, and Nginx log. Weight that has separately been endowed to the logs is shown in table 1. We are most interested in Message log, with the weight 6, and the second is Nginx log, with the weight 3. The worst one is Dmesg log, with the weight 1. Weight can be flexibly adjusted according to changes of businesses.

Table 1 log weight of Message, Dmesg, and Nginx

Log name	Message	Dmesg	Nginx
Log weight	6	1	3

The experiment totally takes 10000 logs, and table 2 are the name, type and number of different log items.

Table 2 Articles of each log item

Log item name	Log type	Log number
A	Message	1300
B	Dmesg	4600
C	Nginx	600
D	Nginx	2700
E	Message	500
F	Nginx	300

In this experiment, we have made comparison between EC algorithm and WFIC algorithm, and at the end of the treatment of 10000 logs, the error of the queried results of log frequent items is 0.1%, which is shown in chart 2:

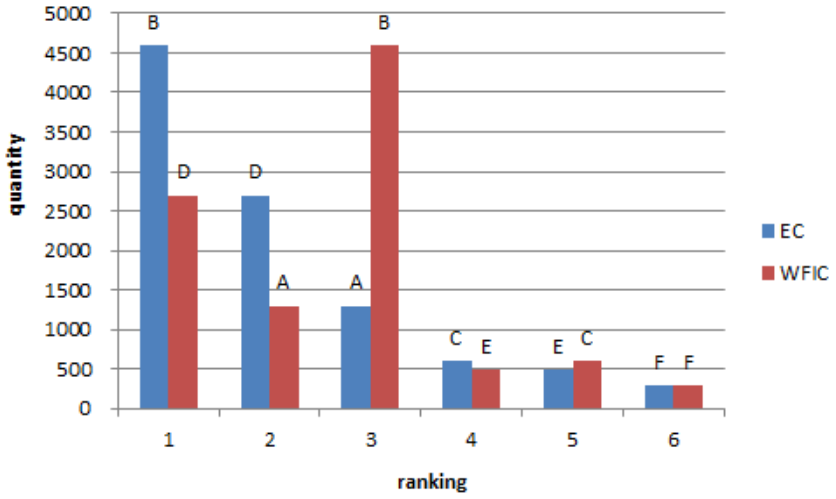

Chart 2 Comparison diagram of ranking result of EC and WFIC algorithms

From chart 2, it can be seen that the ranking result of EC algorithm is B, D, A, C, E, F, and the mining result of frequent items has strictly reflected the quantity of each log item. However, after WFIC algorithm considers log weights of different logs, the ranking result is D, A, B, E, C, F, which verifies the weight allocation of different logs we made before, and it shows that WFIC algorithm is effective.

Summary

Mining frequent items in data stream is a challenging issue. Under the model of data stream management, the algorithm can only scan data once, and the available storage space is limited. The paper has provided with a certain approximation algorithm of ε -— WFIC algorithm. The space cost of this algorithm is $O(\varepsilon^{-1})$, the average processing time of each data item in data stream is $O(1)$. By introducing the concept of log weight, mining results of frequent items in data stream can

better reflect the actual situation, which has important significance to actual production.

References

[1] Cui J,Li T,Lan H．Design and development of the mass data storage platform based on Hadoop [J]．Journal of Computer Research and Development,2012,49 (S1):12－18．

[2] Babcock AK, Babu S, Datar M. Model and issues in data stream systems. In: Popa L, ed. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Madison: ACM, 2002. 1−16.

[3] Charikar M, Chen K, Farach-Colton M. Finding frequent items in data streams. In: Widmayer P, Ruiz FT, Bueno RM, Hennessy M, Eidenbenz S, Conejo R, eds. Proc. of the Int'l Colloquium on Automata, Languages and Programming. Malaga: Springer-Verlag, 2002. 693−703.

[4] Jin C, Qian W, Sha C, Yu JX, Zhou A. Dynamically maintaining frequent items over a data stream. In: Carbonell J, ed. Proc. of the 2003 ACM CIKM Int'l Conf. on Information and Knowledge Management. New Orleans: ACM Press, 2003. 287−294.

[5] Manku GS, Motwani R. Approximate frequency counts over data streams. In: Bernstein P, Ioannidis Y, Ramakrishnan R, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 346−357.

[6] Demaine E, López-Ortiz A, Munro JI. Frequency estimation of Internet packet streams with limited space. In: Möhring RH, Raman R, eds. Algorithms. ESA 2002, Proc. of the 10th Annual European Symp. Rome: Springer-Verlag, 2002. 348−360.

[7] WANG Wei-Ping, LI Jian-Zhong, ZHANG Dong-Dong. An Efficient Algorithm for Mining Approximate Frequent Item over Data Streams. Journal of Software, Vol.18, No.4, April 2007

[8] Leibiusky J,Eisbruch G, Simonassi D． Getting started with storm［M］．Sebastopol: O'Reilly Media, 2012