ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

Verifying search result correctness of frequently occured item set in Data mining for online Customers

[1] Dr. B R Prasad Babu, [2] Saurabh Sharma, [3] Haroon Ali, [4] Nandini B J, [5]Anusha S Prof. & Head
[1][2][3][4][5] Department of Computer Science and Engineering , RRIT, Chikkabanavara , Bangalore-90 ,

***Abstract*— Cloud computing is popularizing the computing paradigm in which data is outsourced to a third-party service provider (server) for data mining. Outsourcing, however, raises a serious security issue: how can the client of weak computational power verify that the server returned correct mining result? In this paper, we focus on the specific task of frequent item set mining. We consider the server that is potentially untrusted and tries to escape from verification by using its prior knowledge of the outsourced data. We propose efficient probabilistic and deterministic verification approaches to check whether the server has returned correct and complete frequent item sets. Our probabilistic approach can catch incorrect results with high probability, while our deterministic approach measures the result correctness with 100% certainty. We also design efficient verification methods for both cases that the data and the mining setup are updated. We demonstrate the effectiveness and efficiency of our methods using an extensive set of empirical results on real datasets.**

Keywords: **Cloud computing, data mining as a service (DMas), security, result integrity verification.**

I. INTRODUCTION

The increasing ability to generate vast quantities of data presents technical challenges for efficient data mining. Outsourcing data mining computations to a third-party service provider (server) offers a cost-effective option, especially for data owners (clients) of limited resources. This introduces the data-mining-as-a-service (DMaS) paradigm. Cloud computing provides a natural solution for the DMaS paradigm. A few active industry projects, for example, Google's Prediction APIs and Microsoft's Daytona project, provide cloud-based data mining as a service to users.

In this paper, we focus on frequent item set mining as the outsourced data mining task. Informally, frequent item sets refer to a set of data values (e.g., product items) whose number of co-occurrences exceeds a given threshold. Frequent item set mining has been proven important in many applications such as market data analysis, networking data study, and human gene association study. Previous research has shown that frequent item set mining can be computationally intensive, due to the huge search space that is exponential to data size as well as the possible explosive number of discovered frequent item sets . Therefore, for those clients of limited computational resources, outsourcing frequent item set mining to computationally powerful service providers (e.g., the cloud) is a natural solution.

Although it is advantageous to achieve sophisticated analysis on tremendous volumes of data in a cost effective way, end users hesitate to place full trust in cloud computing. This raises serious security concerns. One of the main security issues is the integrity of the mining result. There are many possible reasons for the service provider to return incorrect answers . For instance, the service provider would like to improve its revenue by computing with less resources while charging for more. Since sometimes the mining results are so critical that it is imperative to rule out errors during the computation, it is important to provide efficient mechanisms to verify the result integrity of outsourced data mining computations. In this paper, we focus on the problem of verifying whether the server returned correct and complete frequent itemsets. By correctness, we mean that all Itemsets returned by the server are frequent. By completeness, we mean that no frequent itemset is missing in the returned result.

II. ALGORITHM

1. .APRIORI ALGORITHM:

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

The pseudo code for the algorithm is given below for a transaction database {\displaystyle T} T, and a support

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

threshold of {\displaystyle \epsilon } \epsilon . Usual set theoretic notation is employed, though note that {\displaystyle T} T is a multiset. {\display style C_{k}} C_{k} is the candidate set for level {\display style k} k. At each step, the algorithm is assumed to generate the candidate sets from the large item sets of the preceding level, heeding the downward closure lemma. {\display style count[c]} count[c] accesses a field of the data structure that represents candidate set {\display style c} c, which is initially assumed to be zero. Many details are omitted below, usually the most important part of the implementation is the data structure used for storing the candidate sets, and counting their frequencies.

Ck: Candidate item set of size k
Lk : frequent item set of size k
L1 = {frequent items};
for (k = 1; Lk != ; k++) do begin
Ck+1 = candidates generated from Lk;

for each transaction t in database do
increment the count of all candidates in Ck+1 that are contained in t
Lk+1 = candidates in Ck+1 with min_ support
end
return k Lk;

2. CLUSTERING ALGORITHM:

Clustering is a process of partitioning a set of data (or objects) into a set of meaningful sub-classes, called clusters.
Help users understand the natural grouping or structure in a data set. Clustering: unsupervised classification: no predefined classes.
Algorithmic steps for k-means clustering:
Let X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.

1) Randomly select ‘c’ cluster centers.
2) Calculate the distance between each data point and cluster centers.
3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..
4) Recalculate the new cluster center using: where, ‘ci’ represents the number of data points in ith cluster.
5) Recalculate the distance between each data point and new obtained cluster centers.
6) If no data point was reassigned then stop, otherwise repeat from step 3.

III. MODULE

Product Upload:

The admin wants to upload new product to the cloud, it needs to verify the validity of the cloud and recover the real secret key. We show the time for these two processes Happened in different time periods. They only happen in the time periods when the client needs to upload new product to the cloud. Furthermore, the work for verifying the correctness of the can fully be done by the cloud
.

Product Search:

We can consider the dishonest cloud server as a suspect, the data user as a search data to the server .If the server show the search relevant data. Then the user select and buying the product. After continue the relevant product to show the user side. If the user want buying the product and complaint the irrelevant product. The product search based on index based .The cloud provide the data based on index terms. The relevant product specified for the user frequently buying product of services..

Auditing:

Public auditing schemes mainly focus on the delegation of auditing tasks to a third party auditor (TPA) so that the overhead on clients can be offloaded as much as possible. However, such models have not seriously considered the fairness problem as they usually assume an honest owner against an untrusted CSP. Since the TPA acts on behalf of the owner, then to what extent could the CSP trust the auditing result? What if the owner and TPA collude together against an honest CSP for a financial. In this sense, such models reduce the practicality and applicability of auditing schemes. Tpa check the user remarks of the product to be verify.Then the product to be removed from the list based on number of user putting the negative comments of the products.

A. Figures and Tables

TID Transaction

TID	Transaction
1	{i1; i2; i3; i4}
2	{i1; i2; i4}
3	{i2; i4 }
4	{i1; i2; i3}
5	{i1; i3; i4 }
6	{i4}
7	{i2; i3 }
8	{i1; i2 }

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

(a) Transaction dataset D

Items	Inverted List
i1	1,2,4,5,8
I2	1,2,3,4,7,8
I3	1,4,5,7
I4	1,2,3,5,6

(b) Item-based Inverted index EI

Fig. 2: An example of the dataset and its inverted index

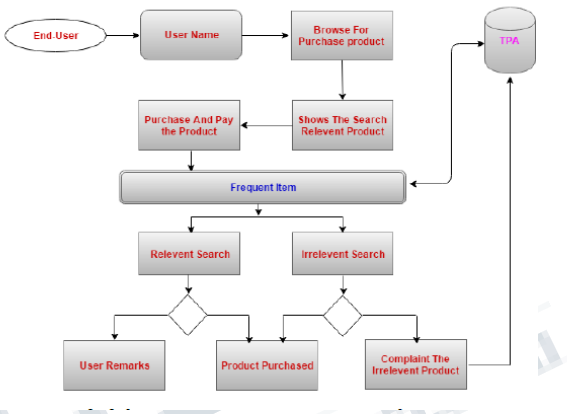
1) Authenticated Data Structure: Before sending the dataset D to the server, the client constructs an authenticated data structure. Before we discuss the details of the authenticated data structure, we first discuss the item-based inverted index. The authenticated data structure will be constructed from the inverted index. In particular, given a dataset D, its item-based inverted index EI consists of a set of inverted lists fL1;L2; : : : ;Lmg, where m is the number of unique items in D. Each inverted list Li 2 EI corresponds to the item Ii in D, and maintains the index of transactions that contains the item Ii. As an example, consider the transaction dataset D shown in Figure 2 (a), Figure 2 (b) shows its item-based inverted index of D.

Now we are ready to discuss how to construct the authenticated data structure. We use the Merkle hash tree T of the inverted index as our authenticated data structure. In particular, the client picks a random value s 2 Z which is kept secret. Then, for each leaf lj of T that corresponds to the j-th inverted list Lj in EI , the client constructs acc(lj) = g , Q , x2Lj (s+x), where g is a generator of the group G1 from an instance of bilinear pairing parameters. Then the client applies a collision-resistant hash function hash( ) recursively over the nodes of T . Each leaf lj of T is assigned the value hj = hash(v1jj : : : jjvwjjacc(lj)), where v1; : : : ; vw are the values in the j-th inverted list Lj that lj corresponds to, while each internal node v with children a and b is assigned to hv = hash(hajjhb). The root of the tree is signed to produce signature sig(EI). The client sends T to the server with D, and keeps sig(EI) locally. The complexity of constructing a Merkle tree of level d1= e levels and m leaves is O(m + H), where 2 (0; 1) is a user-specified constant, and H =Pm j=1 jlj j.

At this point, the client can find all frequent 1-itemsets and infrequent 1-itemsets from the inverted index. It maintains such information for later verification.

2) Verification Procedure: Before outsourcing the dataset D to the server, the client constructs the item-based inverted index EI of D, as well as the Merkle hash tree T of EI . The client keeps the hash value of the root element of T , and sends D and T to the server.

B. Architecture:



C. Probabilistic VS. Deterministic Approaches:

We ran experiments to compare the performance of our probabilistic and deterministic approaches. Table III shows the comparison result on S3 dataset of various settings. We pick the error ratios of 1%, and vary the probabilistic guarantee threshold from 90% to 100% (probability =100% corresponds to our deterministic approach). In general, the deterministic approach brings higher overhead at the server side than the probabilistic approach. However, this is the sacrifice that we have to pay for higher result integrity guarantee. The probabilistic approach fails as it cannot provide required probabilistic correctness guarantee due to the data distribution. The deterministic approach does not have such limit.

IV. CONCLUSION

In this paper, we present two integrity verification approaches for outsourced frequent item set mining. The probabilistic verification approach constructs evidence in frequent item sets. In particular, we remove a small set of items from the original dataset and insert a small set of artificial transactions into the dataset to construct evidence (in)frequent item sets. The deterministic

approaches requires the server to construct cryptographic proofs of the mining result. The correctness and completeness are measured against the proofs with 100% certainty. Our experiments show the efficiency and effectiveness of our approaches. An interesting direction to explore is to extend the model to allow the client to specify her verification needs in terms of budget besides precision and recall threshold..

V. REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), pages 487–499, 1994.

[2] Laszlo Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In STOC, pages 21–32, 1991.

[3] Ran Canetti, Ben Riva, and Guy N. Rothblum. Verifiable computation with two or more clouds. In Workshop on Cryptography and Security in Clouds, 2011.

[4] Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. Power-law relationship and self-similarity in the itemset support distribution: analysis and applications. The VLDB Journal, 17:1121–1141, August 2008.

[5] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In CRYPTO, pages 465–482, 2010.

[6] Fosca Giannotti, Laks V. S. Lakshmanan, Anna Monreale, Dino Pedreschi, and Wendy Hui Wang. Privacy-preserving data mining from outsourced databases. In Computers, Privacy and Data Protection, pages 411–426. 2011.

[7] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal of Computing, 18:186–208, February 1989.

[8] Hakan Hacig¨um¨us¸, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider
model. In SIGMOD, pages 216–227, 2002.