Result Integrity Verification of Outsourced Frequent Itemset Mining

Boxiang Dong, Ruilin Liu, Hui (Wendy) Wang

Stevens Institute of Technology, Hoboken, NJ, USA,
{bdong,rliu3,Hui.Wang}@stevens.edu

Abstract. The data-mining-as-a-service ($DMaS$) paradigm enables the data owner (client) that lacks expertise or computational resources to outsource its mining tasks to a third-party service provider (server). 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 problem of frequent itemset mining, and propose efficient and practical probabilistic verification approaches to check whether the server has returned correct and complete frequent itemsets.

Keywords: Cloud computing, data mining as a service, integrity verification.

1 Introduction

Cloud computing, an emerging trend of provisioning scalable computing services, provides the opportunity that data mining is offered as an outsourced service. Though the data-mining-as-a-service ($DMaS$) paradigm is advantageous to achieve sophisticated data analysis 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. Therefore, it is important to provide efficient mechanisms to verify the result integrity of outsourced data mining computations.

In this paper, we focus on *frequent itemset mining*, an important data mining problem, as the main outsourced data mining service. We aim to address the particular problem of verifying whether the server has 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 server's result.

The key idea of our verification methods is to construct a set of (in)frequent itemsets from *real* items, and use these (in)frequent itemsets as *evidence* to check the integrity of the server's mining result. We remove real items from the original dataset to construct artificial infrequent itemsets, and insert copies of items

that exist in the dataset to construct artificial frequent items. A nice property of our verification approach is that the number of required evidence (in)frequent itemsets is independent from the size of the dataset as well as the number of real frequent itemsets. This is advantageous as our verification approach will be especially suitable for verification of frequent mining on large datasets. Compared with the verification techniques based on fake items (e.g, [13]), our verification techniques are more robust to catch the untrusted server that may try to escape verification by utilizing additional background knowledge such as the item frequency distribution information in the outsourced data. Our experimental results show that our verification approach can achieve strong correctness/completeness guarantee with small overhead.

The paper is organized as follows. We discuss related work in Section 2 and preliminaries in Section 3. We present our EF and EI construction mechanisms for completeness and correctness verification in Section 4 and 5 respectively. In Section 6 we describe the post-processing procedures at the client side. In Section 7, we evaluate the performance of our approach. We conclude in Section 8.

2 Related Work

The problem of verifiable computation was tackled previously by using interactive proofs [4], probabilistically checkable proofs [2], zero-knowledge proofs [14], and non-interactive verifiable computing [3]. Unfortunately, this body of theory is impractical, due to the complexity of the algorithms and difficulty to use general-purpose cryptographic techniques in practical data mining problems.

In the last decade, intensive efforts have been put on the security issues of the database-as-a-service (DaS) paradigm (e.g., [6, 9]). The main focus is the integrity (i.e., correctness and completeness) of result of range query evaluation. Only until recently some attention was paid to the security issues of the data-mining-as-a-service ($DMaS$) paradigm [10]. However, most of these work only focus on how to encrypt the data to protect data confidentiality and pattern privacy, while we focus on integrity verification of mining result.

There is surprisingly very little research [13, 8] on result verification of outsourced data mining computations in the $DMaS$ paradigm. Among these work, [13] is the one the most related to ours. It proposed a result verification scheme for outsourced frequent itemset mining. Its basic idea is to insert some *fake items* that do not exist in the original dataset into the outsourced data; these fake items construct a set of fake (in)frequent itemsets. Then by checking the fake (in)frequent itemsets, the client can verify the correctness and completeness of the mining answer by the server. Though effective, this method assumes that the server has no background knowledge of the items in the outsourced data, and thus it has equal probability to cheat on the fake and real itemsets. We argue that using fake items cannot catch the malicious server that may have some background knowledge of the outsourced dataset. For example, if the server knows that there are k unique items in the original dataset, let $k'(k' > k)$ be the number of items in the outsourced dataset. The probability that an item is

real is k/k'. If the number of artificial items is relatively small compared with the number of real items, the server has a high probability to identify a real item. Furthermore, the verification approach in [13] still preserves the frequency of items, which may enable the server to identify the real/artificial items by the frequency-based attack (e.g, [12, 11]). Our approach is much more challenging than using fake items (as in [13]), since insertion/deletion of real items may modify the true frequent itemsets. Our goal is to minimize the undesired change on the real frequent itemsets, while provide quantifiable correctness/completeness guarantee of the returned result.

3 Preliminaries

Frequent Itemset Mining. Given a transaction dataset D that consists of n transactions, let $\mathcal{I}$ be the set of unique items in D. The *support* of the itemset $I \subseteq \mathcal{I}$ (denoted as $sup_D(I)$) is the number of transactions in D that contain I. An itemset I is *frequent* if its support is no less than a support threshold min_{sup} [1]. The (in)frequent itemsets behave the following two monotone properties: (1) any superset of an infrequent itemset must be infrequent, and (2) any subset of a frequent itemset must be frequent.

Untrusted Server and Verification Goal. Due to many reasons (e.g., code bugs, software misconfiguration, and inside attack), a service provider may return incorrect data mining results. In this paper, we consider the server that possesses the background knowledge of the outsourced dataset, including the domain of items and their frequency information, and tries to escape from verification by utilizing such information. We formally define the correctness and completeness of the frequent itemset mining result. Let F be the real frequent itemsets in the outsourced database D, and F^S be the result returned by the server. We define the *precision* P of F^S as $P = \frac{|F \cap F^S|}{|F^S|}$ (i.e., the percentage of returned frequent itemsets that are correct), and the *recall* R of F^S as $R = \frac{|F \cap F^S|}{|F|}$ (i.e., the percentage of correct frequent itemsets that are returned). Our aim is to catch any answer that does not meet the predefined precision/recall requirement with high probability. Formally, given a dataset D, let F^s be the set of frequent itemsets returned by the server. Let pr_R and pr_P be the probability to catch F^s of recall $R \leq \alpha_1$ and precision $P \leq \alpha_2$, where $\alpha_1, \alpha_2 \in [0, 1]$ are given thresholds. We say a verification method M can verify (α_1, β_1)-*completeness* ((α_2, β_2)-*correctness*, resp.) if $pr_R \geq \beta_1$ ($pr_P \geq \beta_2$, resp.), where $\beta_1 \in [0, 1]$ ($\beta_2 \in [0, 1]$, resp.) is a given threshold. Our goal is to find a verification mechanism that can verify (α_1, β_1)-completeness and (α_2, β_2)-correctness.

4 Construction of Evidence Frequent Itemsets (*EFs*)

Our key idea of completeness verification is that the client uses a set of frequent itemsets as the *evidence*, and checks whether the server misses any evidence frequent itemset in its returned result. If it does, the incomplete answer by the

server is caught with 100% certainty. Otherwise, the client believes that the server returns incomplete answer with a probability. In particular, the probability pr_R of catching the incomplete frequent itemsets F^S of recall R by ℓ evidence frequent itemsets (EFs) is $pr_R = 1-R^\ell$. Clearly, to satisfy (α_1, β_1)-completeness (i.e., $pr_R \geq \beta_1$), it must be true that $\ell \geq \lceil log_{\alpha_1}(1-\beta_1) \rceil$. Further analysis can show that to catch a server that fails to return a small fraction of frequent itemsets with high completeness probability does not need large number of EFs. For instance, when $\alpha_1 = 0.95$ and $\beta_1 = 0.95$, only 58 EFs are sufficient. Apparently the number of required EFs is independent from the size of the dataset as well as the number of real frequent itemsets. Therefore our verification approach is especially suitable for large datasets.

We propose the *MiniGraph* approach to construct EFs. The basic idea of the *MiniGraph* approach is to construct itemsets that are guaranteed to be infrequent in the original dataset D. To construct these itemsets quickly without doing any mining, we construct the itemsets that contain at least one infrequent 1-itemset. The *MiniGraph* approach consists of the following steps:

Step 1: Pick the shortest infrequent itemset (can be 1-itemset) of the largest support as I_s.

Step 2: Find transactions $D_s \subseteq D$ that contain I_s. Construct the *MiniGraph* G from D_s, with the root of G representing I_s, and each non-root node in G representing a transaction in D_s. There is an edge from node N_i to node N_j if the transaction of node N_j is the maximum subset of the transaction of node N_i in D (i.e., no other transactions in D that contain the transaction of node N_i).

Step 3: Mark all nodes at the second level of G as candidates. For each candidate, all of its subset itemsets that contain I_s will be picked as EFs. If the total number of candidates is less than $\ell = \lceil log_{\alpha_1}(1-\beta_1) \rceil$, we add the next infrequent 1-itemset of the largest frequency as another I_s, and repeat Step 1 - 3, until we either find ℓ EFs or there is no infrequent 1-itemset left.

Step 4: For each EF, construct $(min_{sup} - s)$ copies as artificial transactions, where s is the support of EF in D.

The time complexity of the *MiniGraph* approach is $O(|D|)$.

5 Construction of Evidence Infrequent Itemsets (EIs)

Our basic idea of correctness verification is that the client uses a set of infrequent itemsets as the *evidence*, and checks whether the server returns any evidence infrequent itemset. If it does, the incorrect answer by the server is caught with 100% certainty. Otherwise, the client believes that the server returns the incorrect answer with a probability. In particular, the probability pr_P of catching the incorrect frequent itemsets with precision P by using ℓ EIs is $pr_P = 1 - P^\ell$. To satisfy (α_2, β_2)-correctness (i.e., $pr_P \geq \beta_2$), it must satisfy that $\ell \geq \lceil log_{\alpha_2}(1-\beta_2) \rceil$. As pr_P and pr_R (Section 4) are measured in the similar way, we have the same observation of the number of EIs as the number of EFs.

Our EI construction method will identify a set of real frequent itemsets and change them to be infrequent by removing items from the transactions that

contain them. Our goal is to minimize the number of items that are removed. Next, we explain the details.

Step 1: Pick Evidence Infrequent Itemsets (EIs). First, we exclude items that are used as I_s for EF construction from the set of 1-itemset candidates. This ensures that no itemset will be required to be EI and EF at the same time. Second, we insert all infrequent 1-itemsets into the evidence repository $\mathcal{R}$. If $|\mathcal{R}| \geq \ell = \lceil log_{\alpha_2}(1-\beta_2) \rceil$, we terminate EI construction. Otherwise, we compute h, the minimal value to make $\binom{m-|\mathcal{R}|}{h} \geq \ell - |\mathcal{R}|$, where m is the number of unique items in D. Third, we compute k, the minimal value to make $\binom{k}{h} \geq \ell - |\mathcal{R}|$. We pick the first k *frequent* 1-itemsets S following their frequency in ascending order, and construct all h-itemset candidates S_h that contain h items from S. The h-itemset candidates of non-zero support in D will be inserted into $\mathcal{R}$. To efficiently find the itemset I that has non-zero support in D, we make use of a simpler version of the FP-tree [7] to store D in a compressed way. More details of this data structure is omitted due to space limit.

Step 2: Pick Transactions for Item Removal. We aim at transforming those frequent EIs (i.e., artificial infrequent EIs) picked by Step 1, notated as AI, to be infrequent. To achieve this, we pick a set of transactions $D' \subseteq D$, so that for each frequent itemset $I \in AI$, $sup_{D'}(I) \geq sup_D(I) - min_{sup} + 1$.

Step 3: Pick Item Instances for Removal. We decide which items in the transactions picked by Step 2 will be removed. To minimize the total number of removed items, we prefer to remove the items that are shared among patterns in AI. Therefore, we sort items in AI by their frequency in AI in descending order. We follow this order to pick items to be removed.

The time complexity of the EI construction approach is $O(|EI||D| + k!|T|)$, where k is the number of frequent 1-itemsets for construction of EIs, and T is the FP-tree constructed for checking the existence of itemsets in D. Normally $k << m$, where m is the number of items in D, and $|T| << |D|$.

6 Post-Processing

There are two types of side effects by introducing EFs and EIs that need to be compensated: (1) EFs may introduce artificial frequent itemsets that do not exist in D; and (2) EIs may make some real frequent itemsets disappear. Removal of artificial frequent itemsets is straightforward. As the client is aware of the seed item I_s that is contained in all EFs, it only needs to remove all the returned frequent itemsets that contain I_s. To recover missing real frequent itemsets, the client maintains locally all AIs when it constructs EIs. During post-processing, the client adds these AIs back to F^S as frequent itemsets.

7 Experiments

In this section, we experimentally evaluate our verification methods. All experiments were executed on a Macbook Pro machine with 2.4GHz CPU, 4GB mem-

ory, running Mac OS X 10.7.3. We implemented a prototype of our algorithm in Java.

We evaluated our algorithm on two type of datasets: (1) the *dense* dataset in which most of transactions are of similar length, and contain $> 75\%$ of items; and (2) the *sparse* dataset in which the transactions are of skewed length distribution. We use the *NCDC* dataset[1] (500 items, 365 transactions) as the dense dataset, and the *Retail* dataset [2] (16470 items, 88162 transactions) as the sparse dataset. Due to its density/sparsity, $NCDC$ dataset has a large number of frequent 1-itemsets, while *Retail* dataset has a large number of infrequent 1-itemsets. We use the Apriori algorithm [1], a classic frequent itemset mining algorithm, as the main mining algorithm. We use the implementation of Apriori algorithm available at http://www.borgelt.net/apriori.html.

Robustness. We measure the robustness of our probabilistic approach by studying the probability that the incorrect/incomplete frequent itemsets can be caught by using artificial EIs/EFs. We use the *Retail* dataset and vary α_1 and α_2 values to control the amount of mistakes that the server can make on the mining result. For each α_1 (α_2, resp.) value, we randomly modify $(1-\alpha_1)$ ($(1-\alpha_2)$, resp.) percent of frequent (infrequent, resp.) itemsets (including both true and artificial ones) to be infrequent (frequent, resp.). Then with various β_1 and β_2 values, we construct artificial tuples to satisfy (α_1, β_1)-completeness and (α_2, β_2)-correctness. Detection of any missing EF or the presence of any EIs will be recorded as a successful trial of catching the server. We repeat 1,000 times and record the percentage of trials (as *detection probability*) that the server is caught, with $\alpha_1, \alpha_2 \in [0.7, 0.9]$ and $\beta_1, \beta_2 \in [0.7, 0.9]$. It shows that the detection probability for the completeness and correctness verification is always higher than β_1 and β_2 respectively. This proves the robustness of our probabilistic approach. The results are omitted due to limited space.

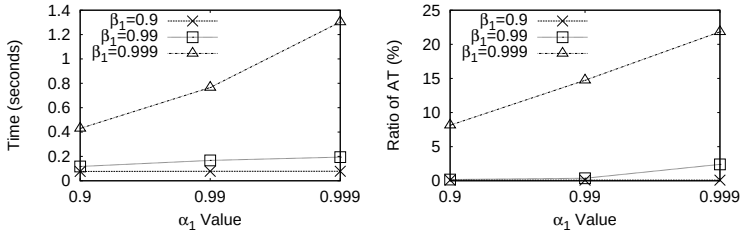
Completeness Verification First, we measure the EF construction time for various α_1 and β_1 values. The result in Figure 1 (a) shows that EF construction time grows when α_1 and β_1 grow, since the *MiniGraph* approach has to search for more I_s to construct more EFs for higher completeness guarantee.

Second, we measure the amount of inserted artificial transactions and compare it with the size of the database. In particular, let t be the number of artificial transactions to be inserted, we measure the ratio $r = \frac{t}{m}$, where m is the number of real transactions in D. As shown in Figure 1 (b), for *Retail* dataset, the inserted artificial transactions only take a small portion of the original database. For example, when $\beta_1 \leq 0.99$, the ratio is less than 3%. Even for large values such as $\alpha_1 = \beta_1 = 0.999$, the ratio is no more than 25% .

Correctness Verification. First, we measure the EI construction time on $NCDC$ dataset. The performance result is shown in Figure 2 (a). It is not surprising that it needs more time to construct EIs for higher α_2 and β_2 values. With a closer look of the result, when $\beta_2 = 0.9$ and 0.99, EI construction is

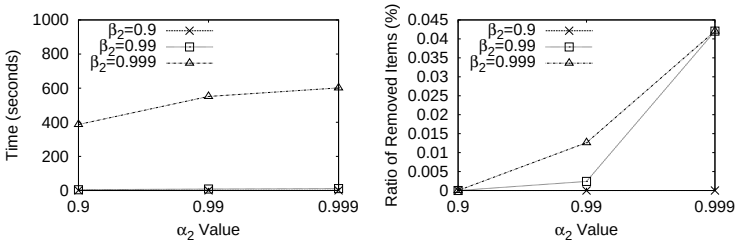
[1] National Climatic Data Center of U.S. Department of Commerce: http://lwf.ncdc.noaa.gov/oa/climate/rcsg/datasets.html

[2] Frequent Itemset Mining Dataset Repository: http://fimi.ua.ac.be/data/.



(a) EF construction time (b) Ratio of Artificial Transactions

Fig. 1. Ratio of Artificial Transactions and Mining Overhead (*Retail* dataset)



(a) EI Construction Time (b) Ratio of Removed Items

Fig. 2. EI Construction ($NCDC$ dataset)

very fast (no more than 1 second), since all EIs are real infrequent itemsets and there is no need to remove any item. However, when β_2 grows to 0.999, the construction time jumps to 400 - 600 seconds, since now the algorithm needs to find frequent itemset candidates to be EIs as well as the items to be removed. We also measure the EI construction time of *Retail* dataset. It does not increase much when β_2 increases from 0.9 to 0.999, since all EIs are real infrequent 1-itemsets.

Second, we measure the amount of item instances that are removed by EI construction. In particular, let t be the number of item instances to be removed, we measure the ratio $r = \frac{t}{|D|}$. The result of $NCDC$ dataset is shown in Figure 2 (b). It can be seen that the number of item instances to be removed is a negligible portion (no more than 0.045%) of $NCDC$ dataset. There is no item that is removed from *Retail* dataset, as it has a large number of infrequent 1-itemsets, which provides sufficient number of EI candidates. This shows that we can achieve high correctness guarantee to catch small errors by slight change of the dataset.

8 Conclusion

In this paper, we present our methods that verify the correctness and completeness of outsourced frequent itemset mining. We propose a lightweight verification approach that constructs evidence (in)frequent itemsets. 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 itemsets. Our experiments show the efficiency and effectiveness of our approach. An interesting direction to explore is to design verification approaches that can provide *deterministic* correctness/completeness guarantee without extensive computational overhead.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of ACM*, 45:501–555, May 1998.
3. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *CRYPTO*, 2010.
4. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, 18:186–208, February 1989.
5. M. T. Goodrich, D. Nguyen, O. Ohrimenko, C. Papamanthou, R. Tamassia, N. Triandopoulos, and C. V. Lopes. Efficient verification of web-content searching through authenticated web crawlers. *PVLDB*, 5(10):920–931, 2012.
6. H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD*, 2002.
7. Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining Knowledge and Discovery*, 8(1):53–87, 2004.
8. R. Liu, H. Wang, A. Monreale, D. Pedreschi, F. Giannotti, and W. Guo. Audio: An integrity auditing framework of outlier-mining-as-a-service systems. In *ECML/PKDD*, 2012.
9. H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying completeness of relational query results in data publishing. In *SIGMOD*, 2005.
10. C.-H. Tai, P. S. Yu, and M.-S. Chen. k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In *SIGKDD*, 2010.
11. W. H. Wang and L. V. S. Lakshmanan. Efficient secure query evaluation over encrypted xml databases. In *VLDB*, 2006.
12. W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis. Security in outsourcing of association rule mining. In *VLDB*, 2007.
13. W. K. Wong, D. W. Cheung, B. Kao, E. Hung, and N. Mamoulis. An audit environment for outsourcing of frequent itemset mining. In *PVLDB*, volume 2, pages 1162–1172, 2009.
14. Yan. Zhu, H. Wang, Z. Hu, G. Ahn, and H. Hu. Zero-knowledge proofs of retrievability. In *Science China Information Sciences*, volume 54, number 8, pages 1608-1617, 2011.