2581-4575

International Journal For Recent Developments in Science & Technology

Crossref

A Peer Reviewed Research Journal

EFFICIENTLY MINING FREQUENT ITEMSETS ON MASSIVE DATA

[1] K.SIDDHARTHA, [2]P.SRINIVAS

[1]Student, Department of Computer Science and Engineering, Kakatiya Institute of Technology & Science, Warangal, T.S,India.

[2]Faculty, Department of Computer Science and Engineering, Kakatiya Institute of Technology & Science, Warangal, T.S,India.

ABSTRACT:

Frequent itemset mining is an important operation to return all itemsets in the transaction table, which occur as a subset of at least a specified fraction of the transactions. The existing algorithms cannot compute frequent itemsets on massive data efficiently, since they either require multiple-pass scans on the table, or construct complex data structures which normally exceed the available memory on massive data. This paper proposes a novel precomputation-based PFIM algorithm to compute the frequent itemsets quickly on massive data. PFIM treats the transaction table as two parts: the large old table storing historical data and the relatively small new table storing newly generated data. PFIM first pre-constructs the quasifrequent itemsets on the old table whose supports are above the lower-bound of the practical support level. Given the specified support threshold, PFIM can quickly return the required frequent itemsets on the table by utilizing the quasi-frequent itemsets. Three pruning rules are presented to reduce the size of the involved candidates. An incremental update strategy is devised to efficiently re-construct the quasi-frequent itemsets when the tables are merged. The extensive experimental results, conducted on synthetic and real-life data sets, show that PFIM has a significant advantage over the existing algorithms and runs two orders of magnitude faster than the latest algorithm..

I.INTRODUCTION

FREQUENT itemset mining is an important operation that has been widely studied in many practical applications, such as data mining [1]–[3], software bug detection [4], spatiotemporal data analysis and biological analysis [5]. Given a transaction table, in which each transaction contains a set of items, frequent itemset mining returns all sets of items whose frequencies (also referred to as support of the set of items) in the table are above a given threshold. Due to its practical importance, since firstly proposed in [6], frequent itemset mining has received extensive attentions and many algorithms are proposed [7]–[9]. The existing frequent itemset mining algorithms can be classified into two groups: candidate-generation-based algorithms [10]–[14] and pattern-growth-based algorithms [15]– [17]. The candidate-generation-based algorithms first generate candidate itemsets and these candidates are validated against the transaction table to identify frequent itemsets. The anti-monotone property is utilized in candidate-generationbased algorithms to prune search space. But the candidate generation-based algorithms require multiple-pass table scans and this will incur a high I/O cost on massive data. The pattern-growth-based algorithms do not generate candidates explicitly. They construct the special tree-based data structures to keep the essential information about the frequent itemsets of the transaction table. By use of the constructed data structures, the frequent itemsets can be computed efficiently. However, pattern-growth-based algorithms have the problem that the constructed data structures are complex and usually exceed the available memory on massive data. To sum up, the existing algorithms cannot compute frequent itemsets on massive data efficiently. In frequent itemset mining, the number of the frequent itemsets normally is sensitive to the value of the support threshold. If the support threshold is small, there will be a large number of frequent itemsets and it is difficult for the users to make efficient decisions. On the contrary, if the support threshold is large, it is possible that no frequent itemsets can be discovered or the interesting itemsets may be missed. Therefore, a proper support threshold is crucial for the practical frequent itemset mining and the users often need to perform frequent itemset mining for several times before the satisfactory support threshold is determined. The process often is interactive. On massive data, the existing algorithms often need a long execution time to compute frequent itemsets and this will affect users' working efficiency seriously [18]. The focus of this paper is to find a new efficient algorithm to compute frequent itemsets on massive data quickly. One useful trick, which is adopted to speed up the execution in the existing algorithms, is to reuse the work done in the counting operation of the shorter itemsets for that of the longer itemsets. In this paper, we want to utilize this reuse idea to a much larger degree. In typical massive data applications, with the increasing data volume and the disk I/O bottleneck, data usually is stored in read/append-only mode [19]. Therefore, the overall data set can be divided into two parts: the much larger old data set storing the historical data, and the relative small new data set storing the newly generated data. Based on the description above, this paper devises a new PFIM algorithm (Precomputationbased Frequent Itemset Mining algorithm) on massive data, which utilizes the pre-constructed frequent itemsets on the old data set to return the frequent itemsets quickly. Since the too small value of support threshold will generate too many frequent itemsets, we assume in this paper that there exists a lower-bound ω of the support threshold specified by the users in practical applications. Because of the real/appendonly mode, given the old table TO, PFIM first pre-constructs the frequent itemsets (refer to as quasi-frequent itemsets in this paper) whose supports are no less than ω. The new transactions are accumulated in the new table TΔ. Taking advantage of the pre-constructed quasi-

frequent itemsts, given the specified support threshold, PFIM can compute the frequent itemsets on TO ∪ TΔ quickly. In the process of execution of PFIM, three pruning rules are devised in this paper to reduce the number of candidate frequent itemsets. An incremental update strategy is proposed in this paper to quickly update the quasi-frequent itemsets when TO and TΔ are merged. The extensive experiments are conducted on synthetic and real-life data sets. The experimental results show that, PFIM outperforms the existing algorithms significantly, it runs two orders of magnitude faster than the latest algorithm..

II.EXISTING SYSTEM:

- The existing algorithms for frequent itemset mining can be divided into two groups mainly: candidate-generation-based algorithms and pattern-growth-based algorithms. This section will review the two kinds of algorithms respectively.
- The existing algorithms cannot compute frequent itemsets on massive data efficiently, since they either require multiple-pass scans on the table or construct complex data structures which normally exceed the available memory on massive data.

III.PROPOSED SYSTEM:

- This paper proposes a novel precomputation-based frequent itemset mining (PFIM) algorithm to compute the frequent itemsets quickly on massive data. PFIM treats the transaction table as two parts: the large old table storing historical data and the relatively small new table storing newly generated data. PFIM first pre-constructs the quasi-frequent itemsets on the old table whose supports are above the lower-bound of the practical support level. Given the specified support threshold, PFIM can quickly return the required frequent itemsets on the table by utilizing the quasi-frequent itemsets. Three pruning rules are presented to reduce the size of the involved candidates. An incremental update strategy is devised to efficiently re-construct the quasi-frequent itemsets when the tables are merged. The extensive experimental results, conducted on synthetic and real-life data sets, show that PFIM has a significant advantage over the existing algorithms and runs two orders of magnitude faster than the latest algorithm

IV.IMPLEMENTATION

- **Admin**

  In this module, the Admin has to login by using valid user name and password. After login successful he can perform some operations such as view and authorize users, Adding Categories Sub-Categories, Adding Product Posts for by Selecting Category and Sub-Categories, Viewing Top- K Utility Item Set Keywords, Viewing all Products in terms of Construction of UP-Tree, Viewing all High Utility Item set Mining Products, Viewing All User Search History and Finding Top K Products Results in Chart.

Viewing and Authorizing Users:

In this module, the admin views all users details and authorize them for login permission. User Details such as User Name, Address, Email Id and Mobile Number.

Add Categories, Sub-Categories and Product Posts:

In this module, the admin adds Categories, Sub-Categories and Product Posts. The Product Posts are added by selecting particular category and Sub-Category and Product Details such as, Product Title, Price, Description and Image of that Product.

View all Products with Ranks and Comments:

In this module, the admin can see all the uploaded products with product ranks and comments. The Product details contain Product title, description, price, and image.

The Comment details include commented user, their comment and the date of comment.

User

In this module, there are n numbers of users are present. User should register before performing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user can perform some operations like viewing their profile details, searching for products based on product description, searching products and viewing them in a UP-Tree Format, Viewing Own Search History and Finding Top K Product Item Sets by selecting category and Top K Value.

Viewing Profile Details:

In this module, the user can see their own profile details, such as their address, email, mobile number, profile Image.

Search Products:

In this, the user search for products based on product description. The matched results will be displayed in two ways: Exact Matched and Related Products. Related Products are the products which are not exactly matched for user entered keyword and they are belong to the same category of exactly matched products category.

Search and View Products in UP-Tree Format:

In this, the user search for products based on product description and the matched products will display in a UP-Tree Format. In a Tree there would be three layers. In a first top layer the Category name and in a second layer the Sub-Category Name and in a last layer the Product Title would be shown and user can see the product details by clicking on product name.

V.CONCLUSION

This paper considers the problem of computing frequent itemsets on massive data. It is found that the existing algorithms cannot perform frequent itemset mining on massive data efficiently. This paper utilizes the idea of reusing the work done previously and devises a precomputation-based PFIM algorithm to quickly acquire the frequent itemsets on massive data. The transaction table consists of two part: the large old table and the relatively small new table. By the quasi-frequent itemsets pre-computed on the old table, PFIM can report the frequent itemsets on massive data efficiently. Three pruning rules are proposed in this paper to speed up the execution of PFIM. The incremental update strategy is presented to re-construct the quasi-frequent itemsets quickly when merging the old table and the new table. The extensive experimental results show that PFIM has a significant performance advantage over the existing algorithms.

VI.REFERENCE

[1] A. Ceglar and J.F. Roddick, “Association mining,” ACM Comput. Surv., 38(2):5, 2006.

[2] H. Cheng, X. Yan, J. Han, and P.S. Yu, “Direct discriminative pattern mining for effective classification,” in Proceedings of the 24th International Conference on Data Engineering, April 7-12, 2008, pp. 169–178.

[3] H. Wang, W. Wang, J. Yang, and P.S. Yu, “Clustering by pattern similarity in large data sets,” in Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, June 3-6, 2002, pp. 394–405.

[4] Z. Li and Y. Zhou, “Pr-miner: automatically extracting implicit programming rules and detecting violations in large software code,” in Proceedings of the 10th European Software Engineering Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, September 5-9, 2005, pp. 306–315.

[5] J.T.L. Wang, M.J. Zaki, H. Toivonen, and D.E. Shasha, editors. ”Data Mining in Bioinformatics,” Springer, 2005.

[6] R. Agrawal, T. Imielinski, and A.N. Swami, “Database mining: A performance perspective,” IEEE Trans. Knowl. Data Eng., vol. 5, no. 6, pp.914– 925, 1993.

[7] C.C. Aggarwal, “Data Mining - The Textbook,” Springer, 2015.

[8] C.C. Aggarwal and J. Han, editors, “Frequent Pattern Mining,” Springer, 2014.

[9] J. Han, H. Cheng, D. Xin, and X. Yan, “Frequent pattern mining: current status and future directions,” Data Min. Knowl. Discov., vol. 15, no. 1, pp.55–86, 2007.

[10] R. Agrawal, T. Imielinski, and A.N. Swami, “Mining association rules between sets of items in large databases,” in Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 1993, pp. 207–216.

[11] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” in VLDB’94, Proceedings of 20th International Conference on Very Large Data Bases, 1994, pp. 487–499.

[12] A. Savasere, E. Omiecinski, and S.B. Navathe, “An efficient algorithm for mining association rules in large databases,” in VLDB’95, Proceedings of21th International Conference on Very Large Data Bases, 1995, pp. 432– 444.

[13] M.J. Zaki, “Scalable algorithms for association mining,” IEEE Trans. Knowl. Data Eng., vol. 12, no. 3, pp.372–390, 2000.

[14] M.J. Zaki and K. Gouda, “Fast vertical mining using diffsets,” in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 326–335.

[15] G. Grahne and J. Zhu, “Fast algorithms for frequent itemset mining using fp-trees,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 10, pp.1347–1362, 2005.