Efficient Data Mining for Frequent Itemsets in Dynamic and Distributed Databases

Adriano Veloso, Wagner Meira Jr. [1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
{adrianov, meira}@dcc.ufmg.br

Abstract. *Data Mining is one of the central activities associated with understanding and exploiting the world of digital data. It is the mechanized process of modeling large databases by means of discovering useful patterns. A frequent itemset is a pattern describing a relevant subset of the data, and a collection of frequent itemsets is particularly useful because it is an extremely compact model of the database. Discovering frequent itemsets in large databases is usually a hard computational task, which can be even harder when data is dynamic and distributed. Applying traditional algorithms in such data results in high communication overhead, excessive wastage of CPU and I/O resources, privacy violations, and often does not meet the stringent rapid response times, to essentially an interactive process of exploiting the data. Hence, there is an urgent need for non-trivial algorithms that can effectively mine frequent itemsets in dynamic and distributed databases. Such algorithms are presented in this master thesis.*

1. Introduction

Accurate modeling of data is the ultimate form of data compression and understanding. By modeling/mining large databases, we are essentially bringing them down in size and complexity to a range that humans can analyze and understand. A collection of frequent itemsets is a compact model of the data, which becomes the ultimate portable data store and can serve to quickly navigate and exploit large volumes of data. For this reason, mining frequent itemsets is a core data mining task, and several algorithms were already proposed in the literature [1, 2, 6]. Typical applications include supermarket sales and banking, astronomy, medicine and biology, but in fact, new applications are mentioned often in the daily press. Some of these new applications have introduced an important new dimension to the problem – distributed sources of dynamic data. For example, every day retail chains record millions of transactions coming from different places, popular web sites log millions of hits world wide etc. In such applications, distributed databases are being updated with a new block of data at regular time intervals. For large intervals, we have the common scenario in many distributed data warehouses. For small intervals, we have high-speed data streams arriving from different sources. Mining such data brings unique opportunities, but it also imposes difficult issues, like interactivity, proper use of the distributed resources, and limited access to privacy-sensitive distributed data. Interactivity is often the key for facilitating data understanding. Making proper use of the distributed resources is crucial because lengthy time delay in response to a user request can disturb the flow of human perception and formation of insight. Providing such issues when data is distributed and dynamic is a challenging task because changes in the data invalidate the collection of frequent itemsets, and simply using traditional approaches to build the new collection results in an explosion in the computational resources required.

We present new algorithms for mining frequent itemsets in dynamic/distributed databases. They make use of incremental, approximate and distributed techniques to save resources, and to effectively handle interactions and updates. Additional highlights include low communication overhead, selective updates, and short-term mining. These features allow our algorithms to mine from dynamic data stored in distributed warehouses, to data streams coming from various sources. Parts of this work have appeared in [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. Please, refer to [15] for a wide review of the related work.

2. Mining Frequent Itemsets in Dynamic and Distributed Databases

Let $\mathcal{D}$ be a database with $\mid \mathcal{D} \mid$ transactions, where each transaction contains a set of items. A set of exactly k items is called a $k-$*itemset*, and the *support* of an itemset is the number of transactions in which it occurs. The itemsets that meet a user-specified *minimum support*, $s_{\mathcal{D}}$, are referred to as *frequent* itemsets, and $F_{\mathcal{D}}$ is the collection of frequent itemsets obtained when mining $\mathcal{D}$. A frequent itemset is *maximal* if it is not subset of any other frequent itemset. Using $\mathcal{D}$ as a starting point, a set of new/old transactions d^+/d^- is added/removed, forming Δ=($\mathcal{D} \cup d^+$)$-d^-$. If a frequent itemset in $\mathcal{D}$ remains frequent in Δ it is called a *retained* itemset. Also, Δ can be divided into n partitions, where each partition δ_i is assigned to site $\mathcal{S}_i$. Let $C.sup$ and $C.sup_i$ be the respective support of itemset C in Δ (global support) and in δ_i (local support). C is *global frequent* if $C.sup \geq s_{\Delta} \times \mid \Delta \mid$; correspondingly, C is *local frequent* in δ_i, if $C.sup_i \geq s_{\Delta} \times \mid \delta_i \mid$. The set of all maximal global frequent itemsets is denoted as MFI_{Δ}, and the set of maximal local frequent itemsets in δ_i is denoted as MFI_{δ_i}. The task of mining frequent itemsets in distributed and dynamic databases is to find F_{Δ}. Before presenting our algorithms and their evaluation, we highlight two important considerations:

1. Summarized Presentation: Due to space constraints, all algorithms will be described in a summarized way, focusing only their main features. Similarly, the experiments showed in this paper are just a small sample of the main results obtained. Please, refer to [5, 6, 7, 8, 9, 10, 11, 12, 13] and to [15] for detailed explanations and the complete set of results and comparisons. The master thesis and relevant papers are available from `http://www.dcc.ufmg.br/~adrianov`.
2. Experimental Setup: Our evaluation was carried out on a cluster of 8 PENTIUM III 1Ghz nodes with 1GB of main memory, interconnected via the MYRINET. Each database partition is already distributed between the nodes, and that each node has only access to its partition of the database. Real databases were used to evaluate the performance of our algorithms. The WCup database is generated from the click-stream log of the 1998 World Cup web site. This database contains 7,618,927 transactions. The VBook database is generated from a 2-month web log from a large electronic bookstore. In this case, each transaction contains the set of books bought by a customer during a single visit to the bookstore.

Mining Frequent Itemsets in Dynamic Databases – Our algorithm, I−ZIGZAG, employs an efficient search which incrementally[1] finds MFI_{Δ} by using $F_{\mathcal{D}}$. Having MFI_{Δ} is sufficient to know which itemsets are frequent in Δ [2]; their supports are then obtained by examining d^+/d^- and using $F_{\mathcal{D}}$, or, where this is not possible, by examining Δ. I−ZIGZAG first verifies if the candidate is a retained itemset. If so, its support is

[1] Incremental algorithms essentially re-use previously mined information (i.e., $F_{\mathcal{D}}$) and try to combine this information with the fresh/obsolete data to update the collection of frequent itemsets.

computed by using d^+/d^- and $F_{\mathcal{D}}$. I−ZIGZAG has two distinguishable features. The first feature comes from the fact that practical data mining is highly iterative and interactive. The user typically changes the parameters and runs the algorithm many times (replicating work). I−ZIGZAG offers the flexibility to change the parameters across mining operations, without work replication [9, 15]. The second feature, short-term mining, keeps the model coherent with the most recent data (desirable when older patterns are no longer relevant). Please, refer to [9] for a complete description of I−ZIGZAG and its features.

Despite the efficiency of I−ZIGZAG, there are situations where the rate in which transactions are added/removed is so high (i.e., data streams) that incremental approaches are ineffective. To deal with such situations we developed WAVE [10], an algorithm which extracts historical trends associated with the support of each itemset. By using such trends, WAVE is able to decide which itemsets need to be updated (i.e., selective updates), and it also uses such trends to estimate the support associated with those itemsets. The approximate result is generated at a small fraction of the total time that is needed to generate an exact result. Please, refer to [8, 10] for a complete description of WAVE.

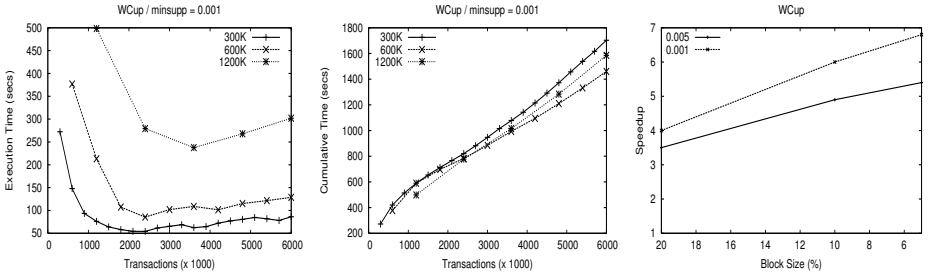

Figure 1: I−ZIGZAG: (a) Total, (b) Cumulative Execution Time , and (c) Speedup.

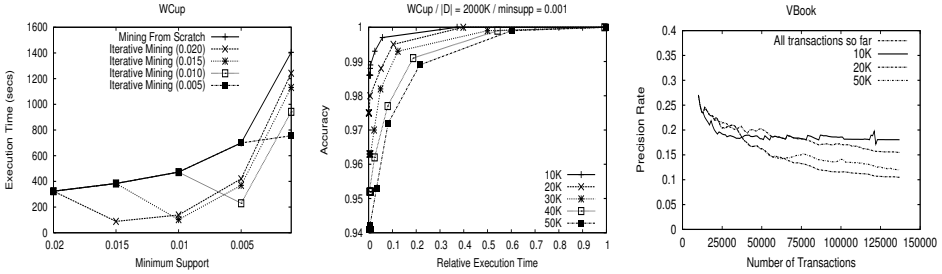

Figure 2: (a) Interactions, (b) Selective Updates, and (c) Short-Term Mining.

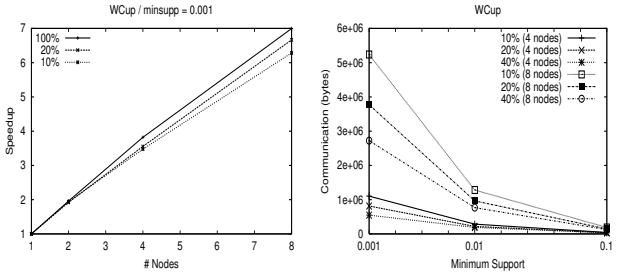

Figure 3: D−ZIGZAG: (a) Speedup, and (b) Communication.

Figure 1(a) shows the execution time results obtained by applying different block sizes. This experiment mimics the reality in practical warehouses, where a large block of data is periodically added to the database (i.e., a block of 300,000 transactions is collected in 6 days). As expected, smaller times are observed for smaller block sizes, but the higher update rate may result in higher overall costs, as can be seen in Figure 1(b). Figure 1(c) shows the speedup numbers of I−ZIGZAG (the speedup is in relation to mining

Δ from scratch), which are inversely proportional to the block size. We also investigated the effectiveness of the interactive features. Figure 2(a) shows the results regarding parameter change operations. We varied s_Δ from 0.020 to 0.001 (there are no data updates). We compared the iterative mining operation against the base-line approach, where Δ is mined from scratch. In this experiment I−ZIGZAG uses $F_\mathcal{D}$ to incrementally find the other collections of frequent itemsets, providing improvements that range from 20% to 50%. Figure 2(b) shows the results concerning the effectiveness of performing selective updates. Our evaluation is based in the accuracy/execution time and we employed small block sizes to simulate high-speed data streams (i.e., a block of 10,000 transactions is collected in 4 hours). Basically, better results are obtained using smaller blocks. Figure 2(c) demonstrates the effectiveness of short-term mining through an actual application which goal is to predict the books to be purchased by the customers. Here, the books selected by a customer while performing a transaction, form an itemset. For each possible prediction, we check the correlation between it and the appraised itemset, and predict the most correlated book. This approach evaluates how the model is coherent with recent data. We employed 3 window sizes, and both $\mid d^- \mid$ and $\mid d^+ \mid$ was set to 5,000 transactions (i.e., approximately one week). We compared the precision using each window size against the base case where there is no windowing in effect. From Figure 2(c) we can see that the best window is the one with the smallest fixed size. Finally, we also compared I−ZIGZAG against the existing state-of-the-art incremental algorithm, ULI [3][2]. I−ZIGZAG was shown to perform much better than ULI, mainly due to the efficient hybrid search employed (i.e., bottom up and top down). The results are reported in [9].

***Lemma 1* − A global frequent itemset is local frequent in at least one partition** [15].
***Lemma 2* − $\bigcup_{i=1}^{n}$ MFI$_{\delta_i}$ determines all global frequent itemsets** [13].

Mining Frequent Itemsets in Distributed Databases − Our algorithm, D−ZIGZAG, works in four steps. In the first step, each site $\mathcal{S}_i$ independently performs an incremental search for MFI$_{\delta_i}$, using I−ZIGZAG on its partition δ_i. Finishing this step, each site $\mathcal{S}_i$ exchanges MFI$_{\delta_i}$ with the other sites, and then they join all local maximal frequent itemsets. At this point each site knows the set $\bigcup_{i=1}^{n}$ MFI$_{\delta_i}$, which according to Lemma 1 is an upper bound (superset) of MFI$_\Delta$ [13]. Next, each site independently performs a top down incremental enumeration of the potentially global frequent itemsets, as follows. Each itemset in $\bigcup_{i=1}^{n}$ MFI$_{\delta_i}$ is broken into k subsets of size $(k-1)$. This process iterates generating smaller subsets and incrementally computing their supports until there are no more subsets to be checked. At the end of this step, all sites have the same set of potentially global frequent itemsets. The final step makes a sum reduction on the local supports of each itemset. The process starts with site $\mathcal{S}_1$, which sends its local supports (i.e., $C.sup_1$ for all $C \in F_{\delta_1}$) to $\mathcal{S}_2$. $\mathcal{S}_2$ sums the support of each itemset with the value of the corresponding itemset obtained from $\mathcal{S}_1$, and sends the result to $\mathcal{S}_3$. This process continues until $\mathcal{S}_n$ has the global supports of all potentially global frequent itemsets. Then $\mathcal{S}_n$ finds all itemsets with global support no less than $s_\Delta \times \mid \Delta \mid$, which constitutes F_Δ (by Lemma 2). This way of communicating is not secure, since each site will have access to the local supports from other sites. Please refer to [5, 11, 13] for a complete description of D−ZIGZAG, and to [12] for a description of a privacy-preserving communication mechanism. Figure 3(a) shows the speedup numbers of D−ZIGZAG, obtained for different parallel (1 to 8 nodes)

[2]ULI was shown to be superior than the previously proposed algorithm, FUP [4]. The source code for ULI was kindly provided to us by its authors.

and incremental configurations (i.e., a block size of 10% means that $\mid \mathcal{D} \mid = 0.9\times \mid \Delta \mid$ and $\mid d^+ \mid = 0.1\times \mid \Delta \mid$). As we can see, better numbers are obtained with smaller block sizes. Figure 3(b) shows the number of bytes transferred between nodes when we varied s_Δ and $\mid d^+ \mid$. As s_Δ decreases, the number of candidates increases, increasing the communication between nodes. Finally, we also compared D−ZIGZAG with other distributed algorithms. D−ZIGZAG is generally better than other approaches, specially in the experiments concerning communication overhead. The results are reported in [11].

3. Conclusions

The basic problem addressed here is the process of mapping (distributed and dynamic) low-level data into other forms that might be more compact, more abstract, or more useful. This process consists of applying incremental and distributed algorithms that efficiently produce particular collection of patterns (or models) over the data. A summary of the main contributions[3] of the thesis was showed in this paper.

References

[1] R. Agrawal et al., Mining Association Rules between Sets of Items in Large Databases, In Proc. of the Int. Conf. on Management of Data, SIGMOD, 207-216, ACM, 1993.

[2] M. Zaki et al., New Algorithms for Fast Discovery of Association Rules, In Proc. of the Int. Conf. on Knowledge Discovery and Data Mining, SIGKDD, 283-290, 1997.

[3] S. Thomaz et al., An Efficient Algorithm for Incremental Updation of Association Rules, In Proc. of the Int. Conf. on Knowledge Discovery and Data Mining, SIGKDD, 263-266, 1997.

[4] D. Cheung et al., Maintenance of Discovered Association Rules in Large Databases: Incremental Updating Technique, In Proc. of the Int. Conf. on Data Eng., ICDE, 86-96, 1996.

[5] M. Otey and A. Veloso and W. Meira, Mining Frequent Itemsets in Distributed and Dynamic Databases, In Proc. of the Int. Conf. on Data Mining, ICDM, IEEE, 617-620, 2003.

[6] A. Veloso et al., Real World Association Mining, In *Advances in Databases*, BNCOD, Lecture Notes in Computer Science: 2405, 77-89, Springer, 2002.

[7] A. Veloso et al., Geração Eficiente de Regras de Associação em Bases de Dados Dinâmicas, In Proc. of the Brazilian Computer Society Conference, SBC, Florianópolis, Brazil, July, 2002.

[8] A. Veloso et al., Mining Reliable Models of Associations in Dynamic Databases, In Proc. of the Brazilian Symp. on Databases, SBBD, 263-277, Brazil, October, 2002.

[9] A. Veloso et al., Mining Frequent Itemsets in Evolving Databases, In Proc. of the Int. Conf. on Data Mining, SDM, April, SIAM, 31-41, 2002.

[10] A. Veloso et al., Efficiently Mining Approximate Models of Associations in Databases, Principles of Data Mining and Knowledge Discovery, PKDD, 435-448, 2002.

[11] A. Veloso et al., New Parallel Algorithms for Frequent Itemset Mining in Large Databases, In Proc. of the Symp. on Comp. Arch. and High Perf. Comp., SBAC, IEEE, 158-166, 2003.

[12] A. Veloso et al., Efficient, Accurate and Privacy-Preserving Mining for Frequent Itemsets in Distributed Databases, In Proc. of the Braz. Symp. on Databases, SBBD, 281-292, 2003.

[13] A. Veloso et al., Parallel and Distributed Frequent Itemset Mining on Dynamic Datasets, In Proc. of the High Perf. Comp. Conf., HiPC, India, IEEE/ACM-SIGARCH, 184-193, 2003.

[14] A. Veloso et al., Parallel and Distributed Methods for Incremental Frequent Itemset Mining, In *Transactions on Systems, Men and Cybernetics*, IEEE, 2004.

[15] A. Veloso, Efficient Data Mining for Frequent Itemsets in Dynamic and Distributed Databases, Master Thesis, December, 2003.

[3] This thesis is extremely multi-disciplinary, encompassing areas such as data mining algorithms and applications [6, 8, 9, 11, 14], statistics [8], databases [12], parallel and distributed systems [5, 11] etc. This work resulted in the publication of 18 papers/articles in major journals/conferences (Adriano is the first author in 16 of them). Some of these papers [7, 8] have won *best paper award*, and *best paper finalist* [12].