Information Sciences 293 (2015) 146–162

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Scalable out-of-core itemset mining

Elena Baralis, Tania Cerquitelli [*], Silvia Chiusano, Alberto Grand

*Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy*

ARTICLE INFO

*Article history:*
Received 3 December 2013
Received in revised form 12 June 2014
Accepted 26 August 2014
Available online 18 September 2014

*Keywords:*
Itemset mining
Data mining

ABSTRACT

Itemset mining looks for correlations among data items in large transactional datasets. Traditional in-core mining algorithms do not scale well with huge data volumes, and are hindered by critical issues such as long execution times due to massive memory swap and main-memory exhaustion. This work is aimed at overcoming the scalability issues of existing in-core algorithms by improving their memory usage. A persistent structure, VLDBMine, to compactly store huge transactional datasets on disk and efficiently support large-scale itemset mining is proposed. VLDBMine provides a compact and complete representation of the data, by exploiting two different data structures suitable for diverse data distributions, and includes an appropriate indexing structure, allowing selective data retrieval. Experimental validation, performed on both real and synthetic datasets, shows the compactness of the VLDBMine data structure and the efficiency and scalability on large datasets of the mining algorithms supported by it.

## 1. Introduction

Itemset mining is an exploratory data mining technique, widely employed to discover valuable, non-trivial correlations between data items in a transactional dataset. The first attempt to perform itemset mining [3] was focused on discovering frequent itemsets, i.e., patterns whose observed frequency of occurrence in the source data is above a given threshold. Frequent itemsets find application in a number of real-life contexts, such as market basket data [3], recommendation systems [19], and telecommunication networks [20]. Frequent itemset mining algorithms have traditionally addressed time scalability, with increasingly efficient solutions that limit the combinatorial complexity of this problem by effectively pruning the search space. To efficiently extract knowledge, most algorithms exploit ad hoc data structures that greatly rely on the available physical memory. However, while the size of real-world databases steadily experiences an exponential growth, mining algorithms are still lagging behind, yielding poor CPU utilization and massive memory swap, thus significantly increasing execution time, and facing the serious bottleneck of main memory. In spite of the increasing availability of physical memory in modern systems and CPUs, the continuous increase in the amounts of analyzed data prompts the need for novel strategies to speed and scale up data mining algorithms. New methods that utilize the secondary storage in the mining process should be the target.

Recently, disk-based extraction algorithms have received an increasing interest. These approaches rely on disk-based data structures to represent the transactional dataset. However, the proposed structures support specific mining algorithms, they typically address specific data distributions and, in general, they often provide only a limited scalability.

[*] Corresponding author. Tel.: +39 011 090 7178; fax: +39 011 090 7099.
*E-mail addresses:* elena.baralis@polito.it (E. Baralis), tania.cerquitelli@polito.it (T. Cerquitelli), silvia.chiusano@polito.it (S. Chiusano), alberto.grand@polito.it (A. Grand).

http://dx.doi.org/10.1016/j.ins.2014.08.073
0020-0255/© 2014 Elsevier Inc. All rights reserved.

This motivates the work described in this paper. This framework, named VLDBMine, includes a persistent transactional data representation and a set of creation and access primitives, to efficiently support large-scale itemset mining. The challenge of this work is to effectively support existing in-core algorithms by enhancing memory usage, thus overcoming scalability issues. The VLDBMine data structure can be profitably exploited to support a variety of state-of-the-art in-core itemset extraction algorithms (e.g., maximal and/or closed itemsets) when the latter outstrip the available memory. Two strategies (loosely- and tightly-coupled) have been proposed to integrate VLDBMine into such mining algorithms, enhancing their scalability. In particular, the tightly-coupled strategy offers the best scalability, by loading, in each mining step, only the data locally required.

VLDBMine is based on a compact disk-based representation, called Hybrid-Tree (HY-Tree), of the *whole* transactional dataset. The HY-Tree exploits two different array-based node structures to adapt its data representation to diverse data distributions. Both structures are variable length-arrays that store different information to compactly represent the dense and the sparse portions of the dataset, respectively. The selection of the node types is automatically driven by the data distribution. VLDBMine also includes an indexing structure, named the Item-Index, which supports selective access to the HY-Tree - portion needed for the extraction task.

The VLDBMine performance has been evaluated by means of a wide range of experiments with datasets characterized by different size and data distribution. As a representative example, VLDBMine has been integrated with LCM v.2 [31], an efficient state-of-the-art algorithm for itemset extraction. The run time of frequent itemset extraction based on VLDBMine is always comparable to or better than LCM v.2 [31] accessing data on a flat file. VLDBMine-based frequent itemset extraction also exhibits good scalability on large datasets.

The paper is organized as follows. Section 2 introduces the VLDBMine data structure, while Section 3 describes its physical organization. Section 4 presents the proposed technique to build VLDBMine on disk. Section 5 discusses the loosely-coupled and tightly-coupled integration strategies, and describes data retrieval techniques to support the data loading phase. Section 6 presents the integration of VLDBMine in the LCM v.2 algorithm. Section 7 discusses how to address the main issues in incrementally updating VLDBMine. The experiments evaluating the effectiveness of the proposed data structure are presented in Section 8. Section 9 reviews existing work in the wide area of frequent itemset mining, focusing on different disk-based solutions proposed in the literature. Finally, Section 10 draws conclusions and presents future developments of the proposed approach.

## 2. The VLDBMine data structure

The VLDBMine persistent representation of the dataset is based on the HY-Tree data structure. The HY-Tree is a prefix-tree-based structure, which encodes the entire dataset and all the information needed to support transaction data retrieval. This tree is hybrid, because two different array-based node structures coexist in it to represent tree nodes and, thus, to reduce the tree size by adapting the data structure to the data distribution. VLDBMine also includes the Item-Index, an auxiliary structure providing selective access to the HY-Tree portion needed for the current extraction task. VLDBMine has been designed to efficiently scale up the itemset mining process on very large transactional datasets. According to the standard definition of transactional datasets, an itemset represents a co-occurrence of items without any temporal ordering of events [18]. A dataset, used as a running example, is reported in Table 1. The corresponding HY-Tree and Item-Index are shown in Figs. 1 and 2, respectively.

### 2.1. The HY-Tree data structure

The HY-Tree has a prefix-tree-like structure. Each transaction is represented by a single path, but a prefix path may represent the common prefix of multiple transactions. In the paths, items are sorted by decreasing values of their global support, given by the number of dataset transactions including each item, and by increasing lexicographic order in case of equal support.

**Table 1**
Example dataset.

| TID | Items | TID | Items |
|---|---|---|---|
| T1 | a, b, e, r, x | T12 | c, e, f, i, o, p, x |
| T2 | h, l, o | T13 | b, d, g, p, x |
| T3 | a, c, g, i, k | T14 | d, p |
| T4 | b, d, e, g, p, v | T15 | b, h |
| T5 | d, j, p | T16 | b, h, l, q |
| T6 | b, i, n, r, s, u | T17 | a, e, j, k, w, x |
| T7 | c, h, z | T18 | a, b, e, r, t, x |
| T8 | a, i, s, t | T19 | b, d, e, m, n, x |
| T9 | h, i | T20 | b, h, q |
| T10 | a, b, i, n, r, z | T21 | h, l, v, w |
| T11 | b, d, e, n, x | | |