PaWI: Parallel Weighted Itemset Mining by means of MapReduce

PaWI: Parallel Weighted Itemset Mining by means of MapReduce

Elena Baralis, Luca Cagliero, Paolo Garza, Luigi Grimaudo
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italy
{*elena.baralis, luca.cagliero, paolo.garza, luigi.grimaudo*}*@polito.it*

Abstract

Frequent itemset mining is an exploratory data mining technique that has fruitfully been exploited to extract recurrent co-occurrences between data items. Since in many application contexts items are enriched with weights denoting their relative importance in the analyzed data, pushing item weights into the itemset mining process, i.e., mining weighted itemsets rather than traditional itemsets, is an appealing research direction. Although many efficient in-memory weighted itemset mining algorithms are available in literature, there is a lack of parallel and distributed solutions which are able to scale towards Big Weighted Data.

This paper presents a scalable frequent weighted itemset mining algorithm based on the MapReduce paradigm. To demonstrate its actionability and scalability, the proposed algorithm was tested on a real Big dataset collecting approximately 34 millions of reviews of Amazon items. Weights indicate the ratings given by users to the purchased items. The mined itemsets represent combinations of items that were frequently bought together with an overall rating above average.

Keywords

H.2.8.b Clustering, classification, and association rules, H.2.8.d Data mining

I. Introduction

The rapid evolution of industrial and scientific technologies prompts modern ICT architectures to deal with massive amounts of data. For example, e-commerce platforms, social network systems, and search engines commonly need for storing and processing terabytes of data. The acquired data are large-scale, heterogeneous, and generated at high speed, thus potentially complex to cope with [1]. For example, to plan promotions and advertisements e-commerce companies continuously need to store and analyze millions of electronic transactions made by their customers as well as their degree of satisfaction about their purchased items.

To support analysts in coping with Big Data there is a compelling need for smart large-scale data mining solutions. Hence, in the last few years an increasing research interest has been devoted to studying and developing supervised and unsupervised data mining algorithms (e.g., classification algorithms [2], and clustering algorithms [3]) that are able to scale towards Big Data. For instance, Mahout [4] and MLlib [5] are notable examples of MapReduce- and Spark-based scalable machine learning and data mining suites.

Frequent itemset mining [6] is an exploratory data mining technique which has largely been used to discover valuable correlations among data items. Frequent itemsets are patterns representing co-occurrences between multiple data items whose observed frequency of occurrence in the source data (the support) is above a given threshold. Frequent itemsets have found application in several application contexts, among which market basket analysis [6], census data analysis [7], and document summarization [8].Since most traditional algorithms (e.g., Apriori [9], FP-Growth [10], Eclat [11]) are not designed to scale towards Big Data, a significant research effort has currently been devoted to developing new large-scale itemset mining algorithms (e.g., [12], [13], [14]).

In real-life applications items are unlikely to be equally important within the analyzed data. For example, items purchased at the market have different prices, medical treatments have different urgency levels, and genes are expressed in biological samples with different levels of significance. Hence, an appealing extension of traditional itemset mining algorithms is to push of item relevance weights into the mining process. To allow treating items/transactions differently based on their relevance in the frequent itemset mining process, the notion of weighted itemset has been introduced [15], [16], [17], [18]. A weight is associated with each data item and it characterizes its local significance within each transaction. Since, to the best of our knowledge, existing large-scale itemset mining algorithms are unable to consider item weights during the mining process, the problem of integrating weights into distributed-based algorithms is challenging.

This paper addresses the problem of extracting frequent itemsets from Big Weighted Datasets. Although there are many in-memory weighted itemset mining algorithms available in literature (e.g., [15], [16], [17], [18]), to the best of our knowledge no parallel and distributed solution for mining frequent weighted itemsets from Big Weighted Data has never been proposed

so far. We propose Parallel Weighted Itemset miner (PaWI), a new parallel and distributed framework to extract frequent weighted itemsets from potentially very large transactional datasets enriched with item weights [19]. The framework relies on a parallel and distributed-based implementation running on an Hadoop cluster [20]. To make the mining process scalable towards Big Data, most analytical steps performed by the system are mapped to the MapReduce programming paradigm [21]. To efficiently perform frequent weighted itemset mining PaWI integrates a variant of the BigFIM algorithm [12] which is able to successfully cope with data enriched with weights. Furthermore, to allow experts to effectively explore the result of the mining process, the proposed system allows us to rank itemsets by (i) weighted support, (ii) traditional support, and (iii) a mix of the above. While the traditional support indicates the generic degree of interest of the considered combination of items, the weighted support integrated in the PaWI framework indicates the average level of interest of the least interesting item within each transaction.

For example, let us consider the frequent combinations of items purchased by e-commerce customers, which companies may deem as interesting to drive promotions and advertising. If for each customer the items in the virtual basket are weighted by a satisfaction rating, analysts can focus their attention only the combinations of items that are frequently bought together with an overall rating above average.

To demonstrate the efficiency and effectiveness of the proposed approach we conducted experiments on a real dataset. Specifically, the PaWI system was tested on a real Big dataset collecting approximately 34 millions of reviews of Amazon items. Weights indicate the ratings given by users to the reviewed items. The mined itemsets, representing combinations of items that were frequently reviewed together with an overall rating above average, were validated by a domain experts. Furthermore, we also experimentally analyzed the scalability of the PaWI system with respect to the number of nodes.

The rest of the paper is organized as follows. Section II overviews most significant related works and compares them with the proposed approach. Section III thoroughly describes the PaWI framework. Section IV experimentally evaluates the effectiveness and efficiency of the proposed approach, while Section V draws conclusions and discusses the future developments of this research.

II. Related work

This paper addresses two complementary research topics: (i) large-scale itemset mining and (ii) weighted itemset mining.

Large-scale itemset mining. Frequent itemset and association rule mining are widely exploratory data mining techniques which were first introduced in [6]. To scale towards large datasets most significant efforts have been devoted to studying parallel and distributed itemset mining strategies. For example, an Apriori-based [9] approach to mining frequent itemsets has been presented in [22]. Since Apriori is known to be less scalable than projection-based and vertical algorithms on complex datasets [10], many attempts to parallelize and distribute different itemset mining strategies have also been made (e.g., [12], [13], [23], [24]). For example, BigFIM is an hybrid algorithm based on MapReduce [21], which combines principles from both Apriori [9] and Eclat [11]. Solutions relying on FP-Growth-like strategies have also been proposed. For example, in [23] the authors exploited a prefix-tree-like structures to drive the parallel itemset mining process. The mining process entails the following steps: first, an horizontal subset of the data is analyzed and a local FP-Tree is built; then the itemset mining process is performed on the local FP-Tree. Finally, the candidate pattern bases from different processing flows are then merged together. In [24] an enhanced strategy for merging processing flows has been proposed, More recently, the Parallel FP-Growth algorithm [13] parallelizes different instances of the recursive FP-Growth process on distributed machines. The key idea is to partition the computation in such a way that each machine executes an independent group of mining tasks thus reducing the communication between machines. To balance the computation load on different machines the authors in [25] proposed to consider the support of singletons, while the works presented in [26] and [27] exploited clustering techniques and data sampling to limit the computational complexity of each mining task. An attempt to discover misleading patterns from Big datasets using MapReduce has been made in [14]. The idea is to compare frequent (unweighted) itemsets mined at different abstraction levels to highlight potentially critical situations. Unlike all the aforesaid approaches, this paper addresses weighted itemset mining instead of traditional itemset mining. To scale towards towards Big Data, the PaWI framework relies parallel and distributed-based implementation running on an Hadoop cluster [20], where most mining steps are mapped to the MapReduce programming paradigm [21].

Weighted itemset mining. In the traditional itemset and rule mining tasks items belonging to each transaction of the source dataset are treated equally. To differentiate items based on their relevance within each transaction, in [15] the authors first addressed the issue of mining more informative association rules, i.e., the Weighted Association Rules (WAR). WARs are association rules enriched with weights denoting item significance. Weights were introduced only during the rule generation step after performing the traditional frequent itemset mining process. To improve the efficiency of the mining process, the authors in [16] pushed item weights deep into the itemset mining process by exploiting the anti-monotonicity of the weighted support measure in an Apriori-based itemset mining process [9]. In [18] a FP-Growth-like weighted itemset mining

Table I
EXAMPLE OF UNWEIGHTED DATASET: ITEM BOUGHT BY CUSTOMERS

Customer id	Purchased items
1	X, Y, Z
2	X, Y, Q
3	X, Y, Z
4	X, Y, Q
5	X, W, Z

algorithm process is presented. Unlike [15], [16], the algorithm proposed in [18] extracts infrequent (rare) itemsets rather than frequent ones. A parallel issue is the extraction of weighted itemsets and rules when coping with data not equipped with preassigned weights. For example, to generate appropriate item weights, in [17] the dataset is modeled as a bipartite hub-authority graph and evaluated by means of a well-known indexing strategy. The PaWI system specifically addresses the problem of *mining frequent weighted itemsets* from data enriched *with preassigned weights*. Unlike [15], [16], [17], [18], it focuses on designing a parallel and distributed approach which is able to cope with large-scale weighted datasets.

III. PARALLEL WEIGHTED ITEMSET MINING FROM BIG DATA

Parallel Weighted Itemset Miner (PaWI) is a new data mining environment aimed to analyze Big Data equipped with item weights. The main environment blocks are briefly introduced below. A more detailed description is given in the following sections.

Data preparation. To prepare the source data to the itemset mining process, data are acquired, enriched with item weights, and transformed using established preprocessing techniques (e.g., data filtering). The result is stored into an HDFS data repository (see Section III-A).

Weighted itemset mining. This step entails the extraction of all frequent weighted and unweighted itemsets from the prepared datasets. To scale towards Big data, the extraction process relies on a parallel and distributed-based itemset miner running on an Hadoop cluster [20] (see Section III-B).

Itemset ranking. To ease the manual exploration of most interesting patterns, the results of the weighted and unweighted itemset mining process are compared with each other and most interesting patterns are selected based on a new quality measure which combines traditional with weighted support counts (see Section III-C).

A. Data preparation

This step entails preparing data to the subsequent itemset mining process. The source data is acquired, stored in a transactional dataset, and equipped with item weights. A *transactional dataset* is a set of transactions [19]. Each *transaction* is a set of (not repeated) *items*. Depending on the context of analysis, items may represent different concepts (e.g., products, objects, places, stocks).

For example, let us consider the dataset reported in Table I. It is an example of (unweighted) transactional dataset consisting of five transactions, each one representing a different customer of a e-commerce company. For each customer the list of purchased items is known. For instance, customer with id 1 bought items X, Y, and Z. Note that each transaction, which represent a distinct electronic basket, may contain an arbitrary number of items.

To consider the relative importance of the items within each transaction during the itemset mining process, items are enriched with weights. A transactional dataset whose items are equipped with weights is denoted as *weighted transactional dataset* [15]. Formally speaking, a weighted transactional dataset is a set of weighted transactions. Each *weighted transaction* is a set of *weighted items*. A weighted item is a pair $\langle$*item*, *weight*$\rangle$, where *weight* is the weight associated with the corresponding *item*.

For example, let us consider the weighted transactional dataset reported in Table II. It extends the traditional transactional dataset in Table I by enriching items with the corresponding weights. More specifically, for each customer the rating (from one to five) given to each purchased item is known. For instance, customer with id 1 rated item X as 3, item Y as 1, and item Z as 5.

The analyzed data are tailored to a weighted transactional data format. Furthermore, if need be, ad hoc preprocessing steps are applied to the raw data to ensure high-quality results. For example, data filtering and discretization are examples of commonly used preprocessing steps [19].

Data filtering entails discarding the items/transactions that are irrelevant for subsequent analyses. For instance, recalling the previous example, duplicate entries of the same customer basket can be removed because they could bias itemset support counts. To ensure the scalability of the knowledge discovery process, the PaWI system performs data filtering as a distributed MapReduce job.

Table II
EXAMPLE OF WEIGHTED DATASET: ITEM RATINGS GIVEN BY CUSTOMERS

Customer id	Purchased items and ratings
1	$\langle X,3\rangle$ $\langle Y,1\rangle$ $\langle Z,5\rangle$
2	$\langle X,2\rangle$ $\langle Y,2\rangle$ $\langle Q,2\rangle$
3	$\langle X,4\rangle$ $\langle Y,2\rangle$ $\langle Z,5\rangle$
4	$\langle X,3\rangle$ $\langle Y,3\rangle$ $\langle Q,2\rangle$
5	$\langle X,2\rangle$ $\langle W,5\rangle$ $\langle Z,4\rangle$

B. Weighted itemset mining

This step focuses on mining frequent weighted itemsets [15] from the prepared weighted dataset. A k-itemset (i.e., an itemset of length k) is a set of k items. The traditional support value of an itemset in a transactional dataset is given by its frequency of occurrence in the source dataset [6].

For example, $\{X,Y\}$ is an itemset indicating the co-occurrence of items X and Y. If we disregard item weights, this itemset has a support equal to 4 in Table I because it occurs in four out of five transactions, meaning that most of the users purchased items X and Y together.

The goal of this paper is to extend traditional large-scale itemset miners to successfully cope with Big weighted data. Hence, for our purposes, the itemset support measure is extended, similar to [15], to the case of weighted data. As previously done in [18], the weighted support of an itemset I in a weighted transactional dataset T is defined as a linear combination of the aggregation weights computed on each transaction in T. Its expression follows.

Definition 1: **Weighted support.** Let D be a weighted transactional dataset, I be a weighted itemset, $\langle i_j, w_j\rangle$ be a arbitrary weighted item such that $i_j \in I$. Let $\mathcal{T}(I)$ be the subset of D's transactions containing all the items in I and f an arbitrary aggregation function defined on item weights. The weighted support of I in D is defined as

$$\text{wsup}\ (I,D) \quad = \quad \sum_{t_q \in \mathcal{T}(I)} f_{j,k,z|i_j,i_k,\ldots,i_z \in I}(w_j, w_k, \ldots, w_z)$$

The weighted support is the summation of all the itemset aggregation weights derived by the aggregation function f for every transaction in T. An arbitrary aggregation function f (e.g., min, max, average, mode) can be potentially applied to aggregate item weights within each transaction. The choice of f depends on the considered use cases. Hereafter, similar to [18], we will consider f=min (i.e., the least weight of any item in I is considered), because, as discussed in Section IV, the selected patterns are deemed as particularly useful for analyzing real Big datasets.

Recalling the running example, let us consider an analysts who would like to discover the combinations of items that were frequently bought together with an overall rating above average. To this aim, he may consider item ratings during support computation by weighting itemset occurrences within each transaction by the least item rating. For instance, recalling the weighted transactional dataset in table for customer with id 1 between X and Y the item with least rating is Y (rating equal to 1), while for customer with id 5 is X (rating equal to 2). Hereafter we will denote as *weighted support* the support of an itemset by considering item weights, whereas as *traditional support* the itemset support disregarding item weights. For instance, $\{X,Y\}$ has weighted support equal to 1+2+2+3+0=8 and traditional support equal to 4.

Given a weighted transactional dataset D and an (analyst-provided) minimum support threshold *minsup*, the PaWI system addresses the extraction of all frequent weighted itemsets from D. To allow comparing weighted itemsets with traditional ones (see Section III-C), PaWI allows experts to mine traditional itemsets as well. As discussed below, the support thresholds enforced during weighted and unweighted itemset mining are potentially different.

The weighted itemset mining process relies on a parallel and distributed-based algorithm running on an Hadoop cluster [20]. To make the mining process scalable towards Big Data, the mining steps are mapped to the MapReduce programming paradigm. MapReduce [21] is a parallel programming framework providing both a relatively simple programming interface together and a robust computational architecture. MapReduce programs consist of two main steps. In the map step, each mapper processes a distinct chunk of the data and produces key-value pairs. In the reduce step, key-value pairs from different mappers are combined by the framework and fed to reducers as pairs of key and value lists. Reducers generate the final output by processing the key/value lists.

To efficiently perform frequent weighted itemset mining with MapReduce PaWI integrates a variant of the BigFIM algorithm [12] which is able to successfully cope with data enriched with weights. The exploitation of weights is challenging because ad-hoc data structures must be used to efficiently maintain the weights associated with each item and transaction by balancing the impacts on computational and communication costs. The following extensions have been proposed:

Distributed transaction splitting. BigFIM relies on two established itemset miners: Apriori [9] and Eclat [11]. We modified the BigFIM algorithm to allow both Apriori and Eclat to successfully cope with weighted data. More specifically, our algorithm generates an equivalent version of the source dataset that includes only transactions with equally weighted items. Let us assume that the weight of an equivalent transaction t_q is w. Then, the occurrence of any itemset in t_q will be weighted by w instead of by 1. Each transaction in the original dataset may correspond to a set of equivalent transactions in the equivalent dataset. A formal definition of the equivalence set of weighted transactions is given in [18]. Note that since two distinct transactions have disjoint equivalent sets the splitting process is straightforwardly parallelizable.

Weighted support counting. Since items are equipped with weights, traditional support counting is replaced with weighted support counting, according to Definition 1. To accomplish itemset support counting different strategies are adopted according to the algorithm used. Specifically, to perform Apriori-based weighted itemset mining, itemset supports are counted by generalizing the word counting problem for documents [21] to weighted itemsets, i.e., each mapper receives a disjoint subset of dataset transactions (i.e., the documents) and reports the items/itemsets (i.e., the words) for which the weighted support count is performed. A reducer combines all partial weighted support counts and reports only the items/itemsets whose weighted support is above the threshold. These frequent weighted itemsets are redistributed to all mappers to act as candidates for the next step of breadth-first search [9] and then the procedure is repeated to mine weighted itemsets of higher length.

To perform Eclat-based weighted itemset mining, each mappers builds the weighted tidlist of the itemsets related to a subset of transactions. The weighted tidlist of an arbitrary item i_j consists of all pairs (*transaction id*, *weight*) such that the transaction related to *transaction id* contains item i_j with weight *weight*. For example, let us assume that a mapper receives the transactions contained in the dataset in Table II. For item Z it generates the following weighted tidlist: {(cid1,5),(cid3,5),(cid5,4)}. The weighted tidlist consists of all pairs (*customer id*, *weight*) for which the transaction related to *customer id* contain item Z with weight *weight*. For instance, the transaction corresponding the electronic basket of the customer with id 1 contains item Z with weight 5. A reducer combines all partial weighted support counts and reports only the items/itemsets whose weighted support is above the threshold. Note that the equivalent transaction weights are not stored in the distributed cache, because Big datasets may potentially consist of millions of transactions.

C. Itemset ranking

The manual exploration of all the itemsets (weighted or not) mined from Big data is practically unfeasible. Hence, to support the knowledge discovery process experts may would like to access only a subset of most interesting patterns. This step focuses on ranking the mined itemsets according to their level of significance in the analyzed data.

To filter and rank the mined itemsets, the support measure is the most commonly used quality index [6]. To cope with weighted data, for each candidate itemset the PaWI system computes both the traditional and weighted support measures. While the traditional support value indicates the observed frequency of occurrence of the considered combination of items in the source dataset, in weighted support counting itemset occurrences are weighted by the least item weight (see Definition 1).

To select itemsets whose average least item weight is maximal the PaWI system combines the weighted and traditional support measure in a new measure called AW-sup, i.e., the *Average Weighted support*. The AW-sup measure is defined as the ratio between the weighted itemset support and the traditional itemset support. It indicates the average per-transaction weight of the least weighted item. Selecting top interesting itemsets based on this measure is potentially interesting in real applications.

For example, let us consider again the example dataset in Table II. According to Definition 1, itemset $\{X,Y\}$ has weighted and traditional support values equal to 8 and 4, respectively. Since transactions represent electronic baskets, the weighted itemset support indicates the overall least item rating computed on the subset of customers who bought both items X and Y, while the traditional support measure indicates the simple frequency of occurrence of the combinations of items in the electronic basket dataset. The AW-sup value of $\{X,Y\}$ is 2, meaning that, on average, for each electronic basket containing items X and Y both items have been rated at least 2. Ranking the mined itemsets by decreasing AW-sup allows experts to consider first the combinations of items that got maximal average ratings. Note that this statistics cannot be straightforwardly computed based on simple averages, because (i) it considers only the electronic baskets containing both X and Y, (ii) for each basket it selects the rating of the least weighted item between X and Y.

Itemsets not satisfying the traditional support threshold are discarded because they represent combinations of items that rarely occur in the analyzed data. The setting of the minimum weighted support threshold is driven by the average rating of the selected items. More specifically, we are interested in exploring the frequent combinations of items with rating above average, i.e., the itemsets whose AW-sup is above a minimum threshold.

IV. EXPERIMENTAL EVALUATION

To assess the effectiveness and efficiency of the proposed approach, we performed a set of experiments on a real dataset [28] collecting approximately 34 millions of Amazon reviews.

Amazon reviews dataset. The Amazon reviews dataset consists of approximately 34 millions of Amazon reviews spanning over a time period of 18 years (from June 1995 to March 2013). The number of users who made at least one review is more than 6 millions and the number of reviewed items is greater than 2 millions. In our context, each transaction consists of a set of items reviewed by a given user. The weight associated with each item is the rating (i.e., the number of stars) given by the user in her/his review. Hence, the transactional version of the Amazon dataset consists of more than 6 millions of transactions. The average transaction length (i.e., the average number of reviews per user) is 4.1. To generate the transactional version of the dataset a MapReduce-based preprocessing step was applied to the original data. Then, a MapReduce job was applied to merge the reviews performed by the same user. More specifically, the reduce created a set of pairs (user identifier, review) and the reducer generated for each user the list of her/his reviews. The itemsets mined from the Amazon dataset represent combinations of items that are frequently rated together with high overall rating. This information could be exploited to improve, for instance, the quality of the traditional *frequently bought together* approach which disregards item ratings.

The experiments were performed on a cluster of 5 nodes running Cloudera's Distribution of Apache Hadoop (CDH5.3). Each cluster node is a 2.67 GHz six-core Intel(R) Xeon(R) X5650 machine with 32 Gbyte of main memory running Ubuntu 12.04 server with the 3.5.0-23-generic kernel. All the reported execution times are real times obtained from the Cloudera Manager web control panel.

The performed experiments aimed at: (i) analyzing the characteristics and usefulness of the mined weighted itemsets (Section IV-A) and (ii) evaluating the scalability of PaWI with respect to the number of nodes (Section IV-B).

A. Result validation

Our approach allows experts to pinpoint combinations of highly rated items purchased together by Amazon customers. To demonstrate the effectiveness of our approach compared to traditional methods, we mined all the itemsets whose AW-sup is above 4 (i.e., average number of stars$>$4) from the Amazon dataset. In the performed experiments we set the traditional minimum support threshold to 0.005% (i.e., the combinations of items reviewed by at least 223 users) and the minimum weighted support threshold to 4·0.005%=0.020% (i.e., the frequent combinations of items whose average rating is above 4). Our assumption is that considering item rates during itemset mining significantly changes the final ranking of the extracted combinations of items.

Table III summarizes the mining result, which consists of 22 highly rated itemsets composted of at least two items each[1], For each itemset we reported (i) the AW-sup, (ii) the itemset raking based on AW-sup (the itemset with top AW-sup ranked first), (iii) the traditional (unweighted) support, and (iv) the raking based on traditional unweighted support (the itemset with top support ranked first). The achieved results confirm that the itemset ranking significantly changes considering the AW-sup measure rather than the traditional support. The ranking based on AW-sup appears to be the most reliable, because it placed first the highly rated combinations of items. To better highlight the difference between the two rankings, the chart in Figure 1 plots the itemset ranking based on traditional support (y-axis) versus the ranking based on AW-sup (x-axis). Figure 1 highlights that the top itemset selected by using the AW-sup measure is ranked 9th if the traditional support is considered. Similar ranking gaps appear for the other itemset rankings. For example, the top ranked itemset based on AW-sup is composed of the first episode or Season 1 and 2 of the Downton Abbey series (see Table III). It indicates that many users watched and the first part of the two Seasons and the overall rating is high (the average number of stars is at least 4.847). This itemset represents an expected situation, because the audience of the first season of a series is likely to watch the first episode of the next season as well. Based on the traditional support measure, this itemset ranked only ninth out of twenty-two. Hence, considering item ratings instead of simple item frequency counts allows us to differentiate between preferred combinations of items and not. Note that itemsets may represent also combinations of three or more items. For instance, {[A Storm of Swords: A Song of Ice and Fire Book III], [A Clash of Kings #392 (A Song of Ice and Fire Ser. Bk. 2)], [A Game of Thrones: A Song of Ice and Fire Book I]} is an example of 3-itemset mined by our approach. These types of correlations are particularly interesting because they often disregarded by traditional data analysis tools (e.g., the original Amazon recommender system considered only the correlations between pairs [29]) or approximated from lower-order correlations among items.

B. Scalability with the number of cluster nodes

We evaluated the scalability of the proposed architecture by measuring the execution time spent while increasing the number of Hadoop cluster nodes. We performed the experiments on a small cluster consisting of 5 nodes. Figure 2 reports

[1]In our analyses we did not consider singletons because we are interested in analyzing the correlations between multiple items.

Table III
FREQUENT ITEMSETS

Itemset	AW-sup	Ranking based on AW-sup	Traditional support	Ranking based on traditional support
{[Downton Abbey: Season 1 Episode 1 "Downton Abbey: Episode 1 - HD": Amazon Instant Video], [Downton Abbey: Season 2 Episode 1 "Original UK Version Episode 1 Part One": Amazon Instant Video]}	4.847	1	333	9
{[Firefly: The Complete Series (2002)], [Serenity (Widescreen Edition) (2005)]}	4.734	2	447	2
{[Measure of a Man], [Bridge Over Troubled Water/This Is The Night]}	4.684	3	244	16
{[Master of Puppets], [Ride the Lightning]}	4.651	4	272	13
{[Harry Potter And The Prisoner Of Azkaban (Turtleback School & Library Binding Edition)], [Hari Potta to kenja no ishi (Harry Potter and the Philosopher's Stone (Japanese Edition)]}	4.623	5	427	5
{[Harry Potter and the Chamber of Secrets], [Hari Potta to kenja no ishi (Harry Potter and the Philosopher's Stone (Japanese Edition)]}	4.599	6	579	1
{[The Lord of the Rings: The Two Towers (Widescreen Theatrical Edition) (2003)], [The Lord of the Rings: The Return of the King - VHS (2003)]}	4.597	7	303	11
{[The Lord of the Rings: The Return of the King - VHS (2003)], [The Lord of the Rings: The Fellowship of the Ring (Two-Disc Widescreen Theatrical Edition) (2002)]}	4.567	8	268	15
{[Abbey Road], [Sgt. Pepper's Lonely Hearts Club Band]}	4.547	9	236	20
{[Harry Potter And The Prisoner Of Azkaban (Turtleback School & Library Binding Edition)], [Harry Potter and the Chamber of Secrets]}	4.546	10	434	4
{[A Storm of Swords: A Song of Ice and Fire Book III], [A Game of Thrones: A Song of Ice and Fire Book I]}	4.526	11	325	10
{[Master of Puppets], [...And Justice For All]}	4.513	12	240	18
{[A Clash of Kings #392 (A Song of Ice and Fire Ser. Bk. 2)], [A Game of Thrones: A Song of Ice and Fire Book I]}	4.494	13	437	3
{[The Lord of the Rings: The Two Towers (Widescreen Theatrical Edition) (2003)], [The Lord of the Rings: The Fellowship of the Ring (Two-Disc Widescreen Theatrical Edition) (2002)]}	4.489	14	405	6
{[Harry Potter And The Prisoner Of Azkaban (Turtleback School & Library Binding Edition)], [Hari Potta to kenja no ishi (Harry Potter and the Philosopher's Stone Japanese Edition)], [Harry Potter and the Chamber of Secrets]}	4.469	15	343	8
{[A Storm of Swords: A Song of Ice and Fire Book III], [A Clash of Kings #392 (A Song of Ice and Fire Ser. Bk. 2)]}	4.459	16	303	12
{[Hari Potta to kenja no ishi (Harry Potter and the Philosopher's Stone Japanese Edition)], [Harry Potter & the Order of the Phoenix]}	4.458	17	240	19
{[A Storm of Swords: A Song of Ice and Fire Book III], [A Clash of Kings #392 (A Song of Ice and Fire Ser. Bk. 2)], [A Game of Thrones: A Song of Ice and Fire Book I]}	4.446	18	224	21
{[Lover Eternal (Black Dagger Brotherhood Book 2)], [Dark Lover (Black Dagger Brotherhood Book 1)]}	4.429	19	224	22
{[Master of Puppets], [Metallica]}	4.239	20	243	17
{[Kill Bill: Vol. 2 - VHS], [Kill Bill: Vol. 1 - VHS (2003)]}	4.123	21	269	14
{[Harry Potter & the Order of the Phoenix], [Harry Potter and the Half-Blood Prince (Book 6)]}	4.112	22	365	7

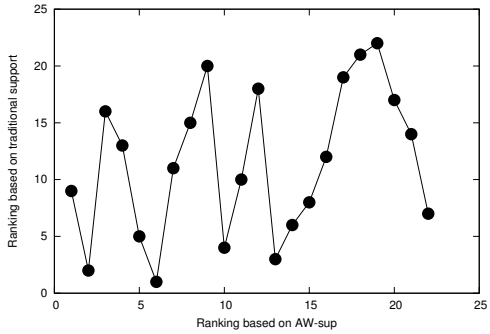

Figure 1. AW-sup vs traditional support.

the execution time achieved by setting the traditional minimum support threshold to 0.005% and the minimum weighted support threshold to 0.020%. The achieved results show that our approach scales roughly linearly in the number of nodes and the speedup approximately corresponds to the number of cluster nodes. The execution time of the proposed algorithm was approximately 8 minutes when all the five nodes were used at the same time. Hence, the execution time was quite small, despite we tested our system on a relatively small cluster. We performed also a set of experiments to evaluate the

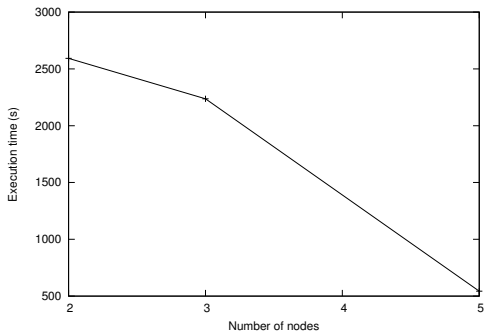

Figure 2. Execution time with respect to the number of cluster nodes on the Amazon dataset.

impact of the minimum support thresholds (weighted and traditional) on the number of mined itemsets and the execution time of the proposed algorithm. The results, not reported here due to the lack of space, meet the expectations, i.e., both the number of itemsets and the execution time increase super-linearly while decreasing the minimum support threshold.

V. Conclusions and future work

This paper presents a parallel and distributed solution to the problem of extracting frequent itemsets from Big Weighted Datasets. The proposed system, running on an Hadoop cluster, overcomes the limitations of state-of-the-art approaches in coping with datasets enriched with item weights. The experiments, performed on a real Amazon dataset, confirm the actionability of the mining result in real context. Future works will entail the application of the proposed approach to recommender systems. For example, discovering combinations of items that were frequently bought together with an overall rating above average could be useful for recommending additional items beyond those already purchased by a given user.

Acknowledgment

The research leading to these results has received funding from the European Union under the FP7 Grant Agreement n. 619633 (Integrated Project ONTIC).

References

[1] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: Current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*, ser. EDBT/ICDT '11. New York, NY, USA: ACM, 2011, pp. 530–533. [Online]. Available: http://doi.acm.org/10.1145/1951365.1951432

[2] H. T. Lin and V. Honavar, "Learning classifiers from chains of multiple interlinked RDF data stores," in *IEEE International Congress on Big Data, BigData Congress 2013, June 27 2013-July 2, 2013*, 2013, pp. 94–101. [Online]. Available: http://dx.doi.org/10.1109/BigData.Congress.2013.22

[3] J. Hartog, R. Delvalle, M. Govindaraju, and M. J. Lewis, "Configuring a mapreduce framework for performance-heterogeneous clusters," in *2014 IEEE International Congress on Big Data, Anchorage, AK, USA, June 27 - July 2, 2014*, 2014, pp. 120–127. [Online]. Available: http://dx.doi.org/10.1109/BigData.Congress.2014.26

[4] "The Apache Mahout Project: The Apache Mahout machine learning library. Available: http://mahout.apache.org/ Last access: March 2015," 2015.

[5] "MLlib: Apache Spark's scalable machine learning library. Available: http://spark.apache.org/mllib/ Last access: March 2015," 2015.

[6] R. Agrawal, T. Imielinski, and Swami, "Mining association rules between sets of items in large databases," in *ACM SIGMOD 1993*, 1993, pp. 207–216.

[7] L. Cagliero and P. Garza, "Itemset generalization with cardinality-based constraints," *Information Sciences*, vol. 244, pp. 161–174, 2013.

[8] E. Baralis, L. Cagliero, S. Jabeen, and A. Fiori, "Multi-document summarization exploiting frequent itemsets," in *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*, 2012, pp. 782–786. [Online]. Available: http://doi.acm.org/10.1145/2245276.2245427

[9] 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*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 1994, pp. 487–499.

[10] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000, pp. 1–12.

[11] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," in *3rd International Conference on Knowledge Discovery and Data Mining (KDD)*, Aug 1997.

[12] S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data," in *SML: BigData 2013 Workshop on Scalable Machine Learning*. IEEE, 2013.

[13] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "Pfp: parallel fp-growth for query recommendation," in *Proceedings of the 2008 ACM conference on Recommender systems*, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 107–114.

[14] E. Baralis, L. Cagliero, T. Cerquitelli, S. Chiusano, P. Garza, L. Grimaudo, and F. Pulvirenti, "Misleading generalized itemset mining in the cloud," in *IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2014, Milan, Italy, August 26-28, 2014*, 2014, pp. 211–216. [Online]. Available: http://dx.doi.org/10.1109/ISPA.2014.36

[15] W. Wang, J. Yang, and P. S. Yu, "Efficient mining of weighted association rules (WAR)," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'00*, 2000, pp. 270–274.

[16] F. Tao, F. Murtagh, and M. Farid, "Weighted association rule mining using weighted support and significance framework," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'03*, 2003, pp. 661–666.

[17] K. Sun and F. Bai, "Mining weighted association rules without preassigned weights," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 4, pp. 489 –495, 2008.

[18] L. Cagliero and P. Garza, "Infrequent weighted itemset mining using frequent pattern growth," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 903–915, 2014. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.69

[19] P.-N. Tan, M. Steinbach, and V. Kumar, *Intoduction to Data Mining*. Addison Wesley, 2005.

[20] "The Apache Hadoop Project. Available: http://hadoop.apache.org/ Last access: March 2015," 2015.

[21] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, ser. OSDI'04, 2004, pp. 10–10.

[22] M. Zaki, "Parallel and distributed association mining: a survey," *Concurrency, IEEE*, vol. 7, no. 4, pp. 14–25, Oct 1999.

[23] O. Zaiane, M. El-Hajj, and P. Lu, "Fast parallel association rule mining without candidacy generation," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, 2001, pp. 665–668.

[24] I. Pramudiono and M. Kitsuregawa, "Tree structure based parallel frequent pattern mining on PC cluster," in *Database and Expert Systems Applications, 14th International Conference, DEXA 2003, Prague, Czech Republic, September 1-5, 2003, Proceedings*, 2003, pp. 537–547.

[25] L. Zhou, Z. Zhong, J. Chang, J. Li, J. Huang, and S. Feng, "Balanced parallel fp-growth with mapreduce," in *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on*, Nov 2010, pp. 243–246.

[26] M. Malek and H. Kadima, "Searching frequent itemsets by clustering data: Towards a parallel approach using mapreduce," in *Web Information Systems Engineering - WISE 2011 and 2012 Workshops - Combined WISE 2011 and WISE 2012 Workshops, Sydney Australia, October 12-14, 2011 and Paphos, Cyprus, November 28-30, 2012, Revised Selected Papers*, 2012, pp. 251–258.

[27] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, "Parma: A parallel randomized algorithm for approximate association rules mining in mapreduce," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12. New York, NY, USA: ACM, 2012, pp. 85–94. [Online]. Available: http://doi.acm.org/10.1145/2396761.2396776

[28] J. J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Seventh ACM Conference on Recommender Systems, RecSys '13*. ACM, 2013, pp. 165–172.

[29] J. Jacobi, E. Benson, and G. Linden, "Personalized recommendations of items represented within a database," Sep. 26 2006, uS Patent 7,113,917. [Online]. Available: http://www.google.com/patents/US7113917