International Journal of Information Sciences and Application.
ISSN 0974-2255 Volume 3, Number 1 (2011), pp. 59-69

http://www.irphouse.com

Algorithms for Mining Sequential Patterns

[1]K.M.V. Madan Kumar and [2]Dr. P.V.S. Srinivas

[1]Research Scholar, CMJ University, Meghalaya, India
[2]Professor & Head, Dept. of CSE,
Geethangali College of Engg. & Tech, Hyderabad, India
E-mail: [1]madankukunuri@gmail.com

Abstract

We are given a database of sequences, where each sequence is a list of transactions ordered by transaction time, and each transaction is a set of items. The problem is to discover all sequential patterns with a user specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern. An example of a sequential pattern is 5% of customers bought `Foundation' and `Ring world' in one transaction, followed by `Second Foundation' in a later transaction''. Here in this paper we have reviewed various algorithms to mine the sequential patterns from sequence data bases and published those algorithms.

Over View of Datamining

Data mining [Chen et al. 1996] is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, XML repository, etc. Also data mining is known as one of the core processes of Knowledge Discovery in Database (KDD).Usually there are three processes in KDD. One is called preprocessing, which includes data cleaning, integration, selection and transformation. The main process of KDD is the data mining process, in this process different algorithms are applied to produce hidden knowledge. After that comes another process called post processing, which evaluates the mining result according to users' requirements and domain knowledge. Regarding the evaluation results, the knowledge can be presented if the result is satisfactory, otherwise we have to run some or all of those processes again until we get the satisfactory result. As not all the data in the database are related to our mining task, the process is to select task related data from the integrated resources and transform them into a format that is ready to be mined. Various data mining techniques are applied to the data source, different

knowledge comes out as the mining result. Those knowledge are evaluated by certain rules, such as the domain knowledge or concepts. After we get the knowledge, the final step is to visualize the results. They can be displayed as raw data, tables, decision trees, rules, charts, data cubs or 3D graphics. This process is try to make the data mining results easier to be used and more understandable.

Types of Mining

Generally speaking, there are two classes of data mining descriptive and prescriptive. Descriptive mining is to summarize or characterize general properties of data in data repository, while prescriptive mining is to perform inference on current data, to make predictions based on the historical data. There are various types of datamining techniques such as association rules, classifications and clustering. Based on those techniques web mining and sequential pattern mining are also well researched. We will review those different types of mining techniques with examples. Association Rule Association rule mining, one of the most important and well researched techniques of data mining, was first introduced in [Agrawal et al. 1993]. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories.

Example 1.1. *In an online book store there are always some tips after you pur- chase some books, for instance, once you bought the book Data Mining Concepts and Techniques, a list of related books such as: Database System 40%, Data Warehouse25%, will be presented to you as recommendation for further purchasing.*

In the above example, the association rules are: when the book *Data Mining Concepts and Techniques* is brought, 40% of the time the book *Database System* is brought together, and 25% of the time the book *Data Warehouse* is brought together. Those rules discovered from the transaction database of the book store can be used to rearrange the way of how to place those related books, which can further make those rules more strong. Those rules can also be used to help the store to make his market strategies such as: by promotion of the book *Data Mining Concepts and Techniques*, it can blows up the sales of the other two books mentioned in the example. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc. Various association mining techniques and algorithms will be briefly introduced and compared later in this Chapter.

Sequential Pattern Mining

As we know, data are changing all the time, especially data on the-web are-highly dynamic. As time passes by, new datasets are inserted, old datasets are deleted while some other datasets are updated. It is obvious that time stamp is an important attribute of each dataset, also it is important in the process of data mining and it can give us more accurate and useful information. For example, association rule mining does not take the time stamp into account, the rule can be Buy A= Buy B. If we take time stamp into account then we can get more accurate and useful rules such as: Buy A

implies Buy B within a week, or usually people Buy A every week. As we can see with the second kind of rules, business organizations can make more accurate and useful prediction and consequently make more sound decisions. A database consists of sequences of values or events that change with time is called a time-series database [Han and Kamber 2000], a time-series database records the valid time of each dataset. For example, in a time-series database that records the sales transaction of a supermarket, each transaction includes an extra attribute indicate when the transaction happened. Time-series database is widely used to store historical data in a diversity of areas such as, financial data, medical data, scientifical data and so on. Different mining techniques have been designed for mining time-series data [Han and Kamber 2000], basically there are four kinds of patterns we can get from various types of time-series data:

Trend analysis: trend analysis is to find the evolution patterns of attributes over time, they can be long-term trend movements, cyclic movements or variations, seasonal movements and irregular/random movements. For example, with the history stock prices of Microsoft, we can model the price changes into a function Y=F(t), which can be illustrated as a time series graph.

Similarity search: similarity search tries to find sequences that differ only slightly. Similarity searching is a blurry matching process that can tolerate some differences within a certain threshold. Based on the length of sequences we are trying to match, sequence matching can be classified as: subsequence matching and whole sequence matching. Suppose we transform the data of stock prices into curves, those curves include many different shapes such as *up, sharp up, down, big down*.

Sequential patterns: sequential pattern mining is trying to find the relationships between occurrences of sequential events, to find if there exist any specific order of the occurrences. We can find the sequential patterns of specific individual items, also we can find the sequential patterns cross different items. Sequential pattern mining is widely used in analyzing of DNA sequence.

Periodical patterns: periodical patterns are those recurring patterns in the time-series database, periodicity can be daily, weekly, monthly, seasonal or yearly. Obviously, periodical patterns mining can be viewed as sequential pattern mining by taking the periodical sequences as a set of sequences.

For periodical pattern mining, we have a sequence of data for a very long time, but we can partition them based on periods and get a set of sequences.

Sequential database consists of sequences of ordered events with or without concrete notion of time. Sequential database is a special case of time series database, consequently most researches in sequential pattern mining focus on two main issues. The first issue is sequential pattern mining, aiming at finding the frequently occurred sequences to describe the data or predict future data. The second issue is periodical pattern mining, which can be viewed as sequential pattern mining.

Sequential Patten Mining Problem

Example 2.1. *From a book store's transaction database history, we can find the frequent sequential purchasing patterns, for example 80% customers who brought the book Database Management typically bought the book Data Warehouse and then brought the book Web Information System with certain time gap.*

Sequential pattern mining was first introduced in [Agrawal and Srikant 1995]. In the above example, all those books need not to be brought at the same time or consecutively, the most important thing is the order in which those books are brought and they are bought by the same customer. 80% here represents the percentage of customers who comply this purchasing habit.

Let D be a database of customer transactions, $I=I_1, I_2, \cdots, I_m$ be a set of m distinct attributes called items, T be transaction that includes {customer-id, transaction-time, item-purchased }, s_i be an itemsets, which contains a set of items from I, S be a sequence that consists of an ordered list of itemsets $< s_1, s_2, \cdots, s_n >$. No customer has more than one transaction with the same transaction time. (10, 20) $\cdots$ are itemsets, while <(30)(90)> $\cdots$ are sequences. All the items in the same round bracket have the same transaction time, while there is an order between round brackets within the large angle bracket

Sequential patterns indicate the correlation between transactions while association rule represents intra transaction relationships. In association rule mining of transaction database, the mining results are about which items are brought together frequently, those items must come from the same transaction. While the result of sequential pattern mining are about which items are brought in a certain order by the same customer, those items come from different transactions.

Sequential pattern is a sequence of itemsets that frequently occurred in a specific order, all items in the same itemsets are supposed to have the same transaction-time value or within a time gap. Usually all the transactions of a customer are together viewed as a sequence, usually called customer-sequence, where each transaction is represented as an itemsets in that sequence, all the transactions are list in a certain order with regard to the transaction-time.

Support, a customer support a sequence s if s is contained in the corresponding customer-sequence, the support of sequence s is defined as the fraction of customers who support this sequence. Sequential pattern mining is the process of extracting certain sequential pat- terns whose support exceed a predefined minimal support threshold.

$$\text{Support}(s) = \frac{\text{Noofsupport customers}}{\text{Total Noof customers}}$$

Table I

Customer-id	Transaction-time	Purchased-items
1	Oct 23' 02	30
1	Oct 28' 02	90
2	Oct 18' 02	10, 20
2	Oct 21' 02	30
2	Oct 27' 02	40, 60, 70
3	Oct 15' 02	30, 50, 70
4	Oct 08' 02	30
4	Oct 16' 02	40, 70
4	Oct 25' 02	90
5	Oct 20' 02	90

(a)sorted transaction data

Large Itemsets	Mapped To
(30)	1
(40) (70)	2
(40, 70) (90)	3
	4
	5

(b)Large Itemset Minsupp=40%

Since the number of sequences can be very large, and users have different interests and requirements, to get the most interesting sequential patterns, usually a minimum support is pre-defined by users. By using the minimum support we can prune out those sequential patterns of no interest, consequently make the mining process more efficient.

Customer-id	Customer Sequence	Transformed DB	After Mapping
1	<(30)(90)>	<{(30)}{(90)}>	<{1}{5}>
2	<(10, 20)(30)(40, 60,70)>	<{(30)}{(40)(70) (40,70)}>	<{1}{2,3,4}>
3	<(30, 50, 70)>	<{(30)(70)}>	<{1, 3}>
4	<(30)(40, 70)(90)>	<{(30)}{(40)(70)(40, 70)}{(90)}>	<{1}{2, 3, 4}{5}>
5	<(90)>	<{(90)}>	<{5}>

©Transformed Data

However some sequential patterns that do not satisfy the support threshold are still interesting. Another metric called *surprise* has been introduced to measure the interestingness of those sequences in [Yang et al. 2001], a sequence s is surprising pattern if its occurrence differs greatly from the expected occurrence by treading every items equal. In the *surprise* metric the information gain in information theory was proposed to measure the overall degree of *surprise*,

Sequential pattern mining approaches

There exists a great diversity of algorithms for sequential pattern mining. . In this Section we first introduce some general and basic algorithms for sequential pattern mining, extensions of those algorithms for special purposes, such as multi-dimensional sequential pattern mining and incremental mining are covered later on. Also periodical pattern mining is elaborated as an extension of sequential pattern mining.

General Algorithms for Sequential Pattern Mining. Most of the basic and earlier algorithms for sequential pattern mining are based on the Apriori property proposed in association rule mining .

[Agrawal and Srikant 1994], the property states that any sub-pattern of a frequent pattern must be frequent. Based on this heuristic, a series of Apriori-like algorithms have been proposed: AprioriAll, AprioriSome, DynamicSome in [Agrawal and Srikant 1995] , GSP [Srikant and Agrawal 1996] and SPADE [Zaki 2001]. Later on another a series of data projection based algorithms were proposed, which includes FreeSpan [Han et al. 2000] and PrefixSpan [Pei et al. 2001]. SPADE [Zaki 2001] is a lattice based algorithm, MEMISP [Lin and Lee 2002] is a memory indexing based approach , while SPIRIT [Garofalakis et al. 1999] integrates constraints by using regular expression.

AprioriAll. Sequential pattern mining was first introduced in [Agrawal and Srikant 1995] by Agrawal, three Apriori based algorithms were proposed. Given the transaction database with three attributes *customer-id*, *transaction-tim* e and *purchased-items*, the mining process were decomposed into five phases
(Example of the first three phases are shown in Table I):

Sort Phase: the original transaction database is sorted with customer-id as the major key and transaction time as the minor key, the result is set of customer sequences. Table I(a) shows the sorted transaction data.

L-itemsets Phase: the sorted database is scanned to obtain large 1-itemsets according to the predefined support threshold. Suppose the minimal support is 40%, in this case the minimal support count is 2, the result of large 1-itemsets is listed in Table I(b).

Transformation Phase: the customer sequences are replaced by those large itemsets they contain, all the large itemsets are mapped into a series of integers to make the

mining more efficient. At the end of this phase the original database is transformed into set of customer sequences represented by those large item- sets. For example transactions with customer-id 1 are transformed into customer sequence <(30)(90)>, this transaction contains two large 1-itemsets (30) and (90)as shown in the Transformed DB, while they are mapped to <(1)(5)> according to the map table in Table I(b). Finally the result database is shown in Table I(c).

Sequence Phase: all frequent sequential patterns are generated from the trans- formed sequential database.

Maximal Phase: those sequential patterns that are contained in other super sequential patterns are pruned in this phase, since we are only interested in maximum sequential patterns.

Since most of the phases are straightforward, researches focused on the sequence phase in [Agrawal and Srikant 1995]. AprioriAll was proposed to find the frequent sequential patterns, which is stated as the fourth phase.

AprioriAll is based on the Apriori algorithm in association rule mining, similarly there are two subprocess. The first is to generate those sequences that may be frequent, which is also called candidate sequences. Then the sequential database is scanned to check the support of each candidate to determine frequent sequential patterns according to minimal support. Since the time cost of the second process is determined by the number of passes over the database and number of candidates, most researchers mainly concern about the candidate generation process and the passes over the database.

The candidate generation process is similar to the AprioriGen in [Agrawal and Srikant 1994]. The Apriori property is also used to prune those candidate sequences whose subsequence is not frequent. The difference is that when we generate the candidate by joining the frequent patterns in the previous pass, different order of combination make different candidates. For example: from *{1}*, *{2}* we can generate two candidates *<{1} {2}* > and *<{2} {1}*>, but in association rule mining we only generate *{*1, 2*}*. Obviously the number of candidate sequences double the size of the candidate itemsets in association rule mining during the generation of 2-candidate sequences. Table II shows how to generate candidate 4-sequences by joining large 3-sequences. By scanning the large 3-itemsets, we find that the first itemsets < 1,2, 3 > and second itemsets < 1, 2, 4 > share their first two items, according to the join condition of Apriori they are joined to produce the two candidates <1, 2, 3, 4> < 1, 2, 4, 3> .Similarly other candidate 4-sequences are generated.

The check process is quite straight forward, by scanning the database the support counts of those candidate sequences can be obtained. By comparing them with the support threshold we can get those frequent sequential patterns. In the maximal sequence phase those sequences whose super-sequences are frequent are pruned out, since we are interested only in maximal sequences.

AprioriAll was the first algorithm for sequential pattern mining, it is based on the naive approach of Apriori association rule mining. The main drawback of AprioriAll. is that too many passes over the database is required and too many candidates are

generated. Based on Apriori algorithm for association rule mining, AprioriAll is not so efficient but it is the basis of many efficient algorithm developed later.

Table II: AprioriAll Candidate Generation $L3$ to $C4$

Large 3-sequences	Candidate 4-sequences
< 1, 2, 3>	< 1, 2, 3, 4 >
< 1, 2, 4>	< 1, 2, 4, 3>
< 1, 3, 4 >	< 1, 2, 3, 5 >
<1, 3, 5 >	< 1, 2, 5, 3>
<2, 3, 4 >	

GSP(Generalized Sequential Pattern). GSP [Srikant and Agrawal 1996] is also an Apriori based algorithm for sequential pattern mining, however it integrates with time constraints and relaxes the definition of transaction, also it considers the knowledge of taxonomies. For time constraints, *maximum gap* and *minimal gap* are defined to specified the gap between any two adjacent transactions in the sequence. If the distance is not in the range between *maximum gap* and *minimal gap* then this two can not be taken as two consecutive transactions in a sequence. This algorithm relaxes the definition of transaction by using a sliding window, which means if the distance between the maximal transaction time and the minimal transaction time of those items is no bigger than the sliding window those items can be taken as in the same transaction.

Sequential pattern mining can be defined as [Srikant and Agrawal 1996]: given a sequence data *D*, a taxonomy *T*, user-defined *min-gap* and *max-gap* time constraints, a user-defined sliding *window-size*, to find all sequences whose *support* is greater than the user-defined *minimum support*.

Like other algorithms GSP has two subprocess: candidate pattern generation and frequent pattern generation.

In the candidate generation process, similar to the AprioriGen function in association rule mining [Agrawal et al. 1993] candidate k-sequences are generated based on the large (k-1)-sequences. Given a sequence $s = \langle s_1 , s_2 , \cdots , s_n \rangle$ and subsequence *c*, *c* is a contiguous subsequence of *s* if any of the following conditions hold:

1. C is derived from *s* by dropping an item from either s1 or s*n*.
2. *c* is derived from *s* by dropping an item from an element s_j that has at least 2 items.
3. *c* is a contiguous subsequence of $c^{\cdot}$, and $c^{\cdot}$ is a contiguous subsequence of *s*. The candidate sequences are generated in two steps as shown in Table III.

Join Phase, candidate k-sequences are generate by joining two (k-1)-sequences that have the same contiguous subsequences, when we joining the two sequences the item can be inserted as a part of the element or as a separated element. For example, <(1, 2)(3)> and <(1, 2)(4)> have the same contiguous subsequence <(1, 2)>, based on those candidate 4-sequence <(1, 2)(3, 4)>, <(1, 2)(3)(4)> and <(1, 2)(4)(3)> can be generated.

Prune Phase, those candidate sequences that have a contiguous subsequence whose support count is less than the minimal support are deleted. Also it uses the hash-tree structure [Park et al. 1995] to reduce the number of candidates to be checked in the next phase.

In AprioriAll it is easy to get the support counts of all those candidate sequences since all the sequences in the database are represented by sub-sequences they contain. With the introduction of maximal and minimal gaps, in GSP it is difficult to get the support counts of candidate sequences. The contain test was introduced with two phases: forward phase and backward phase, which are repeated until all the elements are found. Following is the process of checking if the data-sequence *d* contain a candidate sequence *s* :

Forward Phase: we find successive elements of *s* in *d* as long as the difference between the end-time of the element and the start-time of the previous element is less than max-gap, if the difference is more than max-gap, we switch to the backward phase. If an element is not found then sequence *s* is not contained in *d*.

Table III: GSP candidate generation L3 to c4

Large 3-sequences	Candidate 4-sequences after join	Candidate 4-sequences after pruning
<(1, 2) (3)>	<(1, 2) (3, 4)>	<(1, 2) (3, 4)>
<(1, 2) (4)>	<(1, 2) (3, 5)>	
<(1) (3, 4)>	<(1, 2) (3)(4)>	
<(1, 3) (5)>	<(2)(3, 4)(5)>	
<(2) (3, 4)>	<(1, 2) (4)(3)>	
<(2) (3) (5)>		

Backward Phase: we try to pull up the previous element, suppose s_i is the current element and end-time(s_i)=*t*, we check if there exist transactions containing s_{i-1} and their transaction-times are after *t-max-gap*. Since after pulling up s_{i-1} , the difference between s_{i-1} and s_{i-2} may not satisfy the gap constraints, the backward pulls back until the difference of s_{i-1} and s_{i-2} satisfies the max-gap or the first element has

been pulled up. Then the algorithm switches to the forward phase, if any element can not be pulled up the data-sequence *d* does not contain *s*.

The number of rules becomes large when extended sequences are mined because the sequences become more dense, consequently there are many redundant rules. To avoid uninteresting sequences, first the ancestors are pre-computed for each item, ancestors that are not in the candidate sequences are dropped. Secondly they do not count sequential patterns that contain both the item and its ancestors.

For the Apriori based algorithms, the number of candidate sequences is quite large and they require many passes over the whole database as well. But the two approaches are the basis of further researches on mining sequential pattern, at least all the sequential patterns can be generated though they are not so efficient. GSP performs relatively better than AprioriAll, in GSP the number of candidate sequences is much smaller, also time constraints, taxonomies are integrated during the sequential patterns mining process to produce more knowledge.

Lat A GSP-based algorithm called MFS(Mining Frequent Sequences) [Zhang et al. 2001], rather than scan the database many times as the length of the maximal sequence, MFS proposed a two-stage algorithm. First a sample of the database is mined to get the rough estimation of frequent sequences, based on those estimation the database is scanned to check and refine the candidate sequences until no more frequent sequences can be found. The difference between MFS and GSP is the function of candidate generation: in GSP each time only those candidates of the same length are generated by joining those frequent sequences in the previous pass, but in MFS candidates of different lengthes are generated by joining all those known frequent sequences. Experiments shown that MFS generates the same set of frequent sequences as GSP while it outperforms GSP by reducing the I/O cost.

Conclusion

In this paper, we surveyed the list of existing association rule and sequential pattern mining techniques. This investigation is prepared to our new project titled mining *sequential patterns with multiple minimum supports*

References

[1] Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, P. Buneman and S. Jajodia, Eds. Washington, D.C., 207–216.

[2] Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 487–499.

[3] Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, P. S. Yu and A. S. P. Chen, Eds. IEEE Computer Society Press, Taipei, Taiwan, 3–14.

[4] GaM. N., Rastogi, R., and Shim, K. 1999. Spirit: Sequential pattern mining with regular expression constraints. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, Eds. Morgan Kaufmann, 223–234.

[5] Han, J. and Kamber, M. 2000. *Data Mining Concepts and Techniques*. Morgan Kanufmann.

[6] Lin, M.-Y. and Lee, S.-Y. 2002. Fast discovery of sequential patterns by memory indexing. In*Proc. of 2002 DaWaK*. 150–160.

[7] Srikant, R. and Agrawal, R. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, Eds. Vol. 1057. Springer-Verlag, 3–17.

[8] Zhang, M., Kao, B., Yip, C., and Cheung, D. 2001. A gsp-based efficient algorithm for mining frequent sequences. In *Proc. 2001International Conference on Artificial Intelligence (IC-AI 2001)*.

[9] Zheng, Q., Xu, K., Ma, S., and Lv, W. The algorithms of updating sequential patterns.