Frequent Itemset Mining and Association Rules

Susan Imberman
City University of New York, USA

Abdullah Uz Tansel
Bilkent University, Turkey

INTRODUCTION

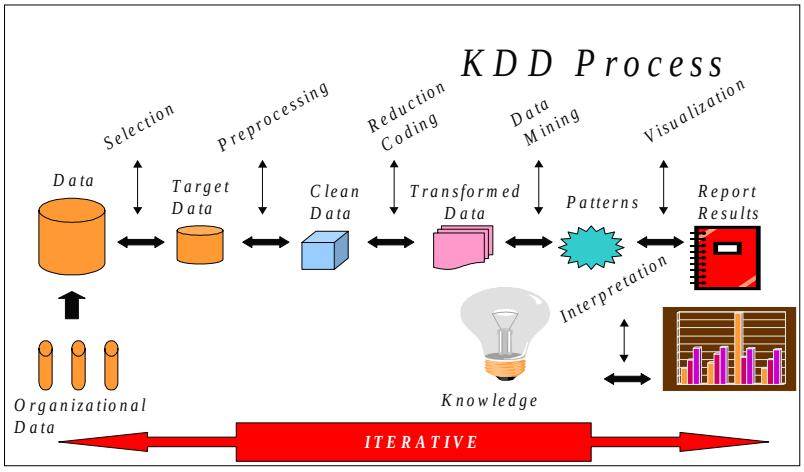
With the advent of mass storage devices, databases have become larger and larger. Point-of-sale data, patient medical data, scientific data, and credit card transactions are just a few sources of the ever-increasing amounts of data. These large datasets provide a rich source of useful information. Knowledge Discovery in Databases (KDD) is a paradigm for the analysis of these large datasets. KDD uses various methods from such diverse fields as machine learning, artificial intelligence, pattern recognition, database management and design, statistics, expert systems, and data visualization.

KDD has been defined as "the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" (Fayyad, Piaetsky-Shapiro, & Smyth, 1996). The KDD process is diagramed in Figure 1.

First, organizational data is collated into a database. This is sometimes kept in a data warehouse, which acts as a centralized source of data. Data is then selected from the data warehouse to form the target data. Selection is dependent on the domain, the end-user's needs, and the data mining task at hand. The preprocessing step cleans the data. This involves removing noise, handling missing data items, and taking care of outliers. Reduction coding takes the data and makes it usable for data analysis, either by reducing the number of records in the dataset or the number of variables. The transformed data is fed into the data mining step for analysis, to discover knowledge in the form of interesting and unexpected patterns that are presented to the user via some method of visualization. One must not assume that this is a linear process. It is highly iterative with feedback from each step into previous steps. Many different analytical methods are used in the data mining step. These include decision trees, clustering, statistical tests, neural networks, nearest neighbor algorithms, and association rules. Association rules indicate the co-occurrence of items in market basket data or in other domains. It is the only technique that is endemic to the field of data mining.

Organizations, large or small, need intelligence to survive in the competitive marketplace. Association rule discovery along with other data mining techniques are tools for obtaining this business intelligence. Therefore, association rule discovery techniques are available in toolkits that are components of knowledge management systems. Since knowledge management is a continuous process, we expect that knowledge management techniques will, alternately, be integrated into the KDD process. The focus for the rest of this article will be on the methods used in the discovery of association rules.

Figure 1. The KDD process



BACKGROUND

Association rule algorithms were developed to analyze market basket data. A single market basket contains store items that a customer purchases at a particular time. Hence, most of the terminology associated with association rules stems from this domain. The act of purchasing items in a particular market basket is called a transaction. Market basket data is visualized as Boolean, with the value 1 indicating the presence of a particular item in the market basket, notwithstanding the number of instances of an item; a value of 0 indicates its absence. A set of items is said to satisfy a transaction if each item's value is equal to 1. Itemsets refer to groupings of these items based on their occurrence in the dataset. More formally, given a set $I = \{ i_1, i_2, i_3, \ldots i_n \}$ of items, any subset of I is called an itemset. A *k-itemset* contains *k* items. Let X and Y be subsets of I such that $X \cap Y = \phi$. An association rule is a probabilistic implication $X \Rightarrow Y$. This means if X occurs, Y also occurs. For example, suppose a store sells, among other items, shampoo (1), body lotion (2), hair spray (3), and beer (4), where the numbers are item numbers. The association rule *shampoo, hair spray* $\Rightarrow$ *beer* can be interpreted as, "those who purchase shampoo and hair spray will also tend to purchase beer."

There are two metrics used to find association rules. Given an association rule $X \Rightarrow Y$ as defined above, the support of the rule is the number of transactions that satisfy $X \cup Y$ divided by the total number of transactions. Support is an indication of a rule's statistical significance. Interesting association rules have support above a minimum user-defined threshold called *minsup*. Given the database represented in Figure 2, the support of the association rule *shampoo, hair spray* $\Rightarrow$ *beer* is equal to the number of transactions where shampoo, hairspray, and beer are equal to 1. This is equal to the shaded region

Figure 2. Support of shampoo, hair spray ⇒ beer 4/12 or 33%

1. Shampoo	2. Hair Spray	3. Body Lotion	4. Beer
1	0	1	1
1	1	1	0
1	1	1	1
1	0	1	1
0	0	0	1
1	0	1	1
1	1	1	0
1	1	1	1
0	1	0	1
1	1	1	1
1	1	1	1
1	0	0	1

Figure 3. Support of shampoo and hair spray

1. Shampoo	2. Hair Spray	3. Body Lotion	4. Beer
1	0	1	1
1	1	1	0
1	1	1	1
1	0	1	1
0	0	0	1
1	0	1	1
1	1	1	0
1	1	1	1
0	1	0	1
1	1	1	1
1	1	1	1
1	0	0	1

and consists of a support of 4 out of 12 transactions, or 33%. Frequently occurring itemsets, called frequent itemsets, indicate groups of items customers tend to purchase in association with each other. These are itemsets that have support above the user-defined threshold, *minsup*.

Given an association rule $X \Rightarrow Y$ as defined above, the confidence of a rule is the number of transactions that satisfy $X \cup Y$ divided by the number of transactions that satisfy X. In Figure 3, the shaded portion indicates the support of Shampoo and Hair Spray. The confidence is then the support of the itemset Shampoo, Hairspray and Beer, divided by the support of Shampoo and Hairspray which equals 4/6 = 66%. It is common practice to define a second threshold based on a user-defined minimum confidence called *minconf*. A rule that has support above *minsup* and confidence above *minconf* is an interesting association rule (Agrawal, Imielinski, Swami, 1993; Agrawal & Srikant, 1994; Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996).

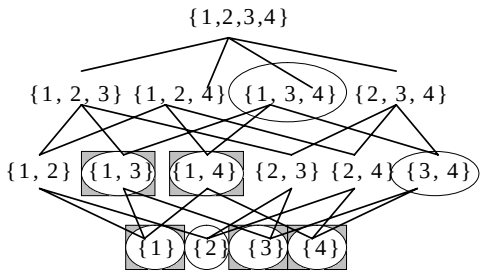
FINDING ASSOCIATION RULES

Finding association rules above *minconf*, given a frequent itemset, is easily done and linear in complexity. Finding frequent itemsets is exponential in complexity and more difficult, thus necessitating efficient algorithms. A brute force approach would be to list all possible subsets of the set of items I and calculate the support of each. Once an itemset is labeled frequent, partitions of the set's items are used to find rules above *minconf*. Continuing our example, assume *minsup* = 65%. Figure 4 lists all the subsets of the set of the items in Figures 2 and 3. The shaded areas indicate the frequent itemsets with support equal to or above 65%. The set of all itemsets forms a lattice, as seen in Figure 5.

Figure 4. Itemsets and their support

Itemset	Support (in percent)	Itemset	Support (in percent)
{1}	83	{2}	58
{3}	75	{4}	83
{1,2}	50	{1,3}	75
{1,4}	66	{2,3}	50
{2,4}	41	{3,4}	58
{1,2,3}	50	{1,2,4}	33
{1,3,4}	58	{2,3,4}	33
{1,2,3,4}	33		

Figure 5. Lattice of itemsets



One can see that the brute force method grows exponentially with the number of items in *I*. In a database containing thousands of items, a brute force approach can become intractable. Algorithms that are exponential relative to the number of variables are said to suffer from "the curse of dimensionality."

If we look at the 1-itemsets in Figure 4, we notice that itemset {2} is below *minsup*. In fact, all supersets of itemset {2} are below *minsup* as well. This is illustrative of the upward closure property of support. If an itemset is not frequent, then the itemset's supersets will not be frequent. Many association rule algorithms use this property to prune the search space for frequent itemsets. Apriori is one such algorithm (Agrawal et al., 1993; Agrawal & Srikant, 1994; Agrawal et al., 1996).

APRIORI ALGORITHM

Apriori uses the upward closure property of support to move level wise through the lattice. To find frequent itemsets, Apriori first scans the data set for the counts of 1-itemsets, since all 1-itemsets are candidates to be frequent. Those frequent 1-itmesets are used to generate the 2-itemsets that are candidates to be frequent. In general, Apriori generates candidate itemsets at a particular level *k* from the *k-1* itemsets at level *k-1*. This is done in the algorithm's join step. If two frequent itemsets at level *k-1* have the same *k-2* items in common, we form the union of these two sets. The resulting set is a candidate *k-itemset*. Each of these *candidate* itemsets are checked to see if any of their subsets are not frequent. If so, they are pruned from consideration in the prune step, since if you recall, supersets of itemsets that are not frequent, are themselves not frequent. Candidate itemsets that do not have support equal to or above *minsup* are also pruned. The algorithm proceeds level wise through the lattice, until there are no more candidate itemsets generated.

Using the data set from Figure 1, in level 1 of Figure 5, the algorithm starts with all the 1-itemsets as candidate itemsets. Candidate itemsets in the figure are circled. Counting the support of each itemset, we see that all but {2} are frequent. A box indicates frequent itemsets. Itemsets {1}, {3}, and {4} combine to form candidate 2-itemsets {1,3}, {1,4}, and {3, 4}. From the data we see that itemsets {1, 3} and {1, 4} are frequent. Since {1, 3} and {1, 4} have *k-1* items in common, these itemsets are combined to form the candidate 3-itemset {1, 3, 4}. Itemset {3, 4} is a subset of {1, 3, 4}. Since {3, 4} is not frequent, {1, 3, 4} cannot be frequent. The algorithm stops since we cannot generate any more candidate itemsets.

VARIATIONS ON APRIORI

Researchers have devised improvements to overcome the bottlenecks in the Apriori algorithm. One bottleneck is the time needed to scan the dataset since the dataset is huge, normally terabytes large. Because of this, a lot of the work done by these algorithms is in searching the dataset. The authors of Apriori realized that transactions that do not contain *k* large itemsets would not contain *k+1* large itemsets. Thus avoiding further scans of the dataset (Agrawal et al., 1996). Another improvement was to implement the use of transaction identification lists (TID lists). These are the lists of transactions an itemset is contained in. The dataset is scanned only once to create the TID lists for the 1-itemsets. The TID lists for itemsets on any level *k+1* is created by taking the intersection of the TID lists of the itemsets from level *k* used in their creation. The problem with TID lists is that initially, the size of the list has the potential to be larger than the dataset. In recognition of this, the authors of Apriori developed Apriori Hybrid, which scans the dataset in the beginning levels of the algorithm, and then switches to TID lists.

Other researchers have taken different approaches to the problem of scanning large datasets. In Dynamic Hashing and Pruning (DHP), it was recognized that in level wise algorithms like Apriori, much of the work is

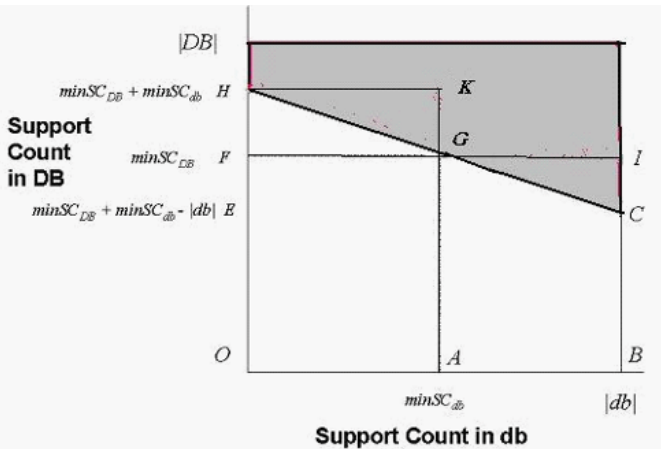
done in generating and counting the 2-itemsets (Park, Chen, & Yu, 1995). The approach here was to hash the candidate 2-itemsets. The number of itemsets in each bin is stored. If the total count of the itemsets in a bin is not larger than or equal to *minsup*, then the itemsets in that bin cannot reach *minsup*. These itemsets are pruned, and the algorithm proceeds as in Apriori.

Another approach was to break the dataset into n partitions such that each partition fits into main memory (Savasere, Oiecninski, & Navathe 1998). The premise is that any global large itemset must also be one of the local frequent itemsets found in a partition. Once frequent itemsets in local partitions are found, the dataset is scanned to determine which of these is global.

Another approach has been to create a random sample from the dataset large enough to fit into memory (Toivonen, 1996). The sample is then used to find frequent itemsets. In order to increase the probability that those itemsets found in the sample would include all frequent itemsets from the dataset, the sample is scanned with a lower support than that used for the dataset. The transactions of the dataset, not in the sample, are then used to check the support counts of the sample frequent itemsets. Thus only one scan of the dataset is required, but there is no guaranty that all frequent itemsets will be found.

Datasets that are increasing in size pose the problem of how to efficiently mine the new data. One could run an algorithm like Apriori on the "new" larger dataset, but this ignores all previous work done in discovering frequent itemsets. In addition it is costly and inefficient since most of the work done in finding frequent itemsets is in scanning the dataset. To avoid redoing one can take an incremental approach whereby you use the information obtained in previous association rule processing to reduce the amount of dataset scans when new transactions are added (Ayan, Tansel, & Arkun, 1999).

Figure 6. The incremental association rule problem



Below, we list the notation used in incremental association rule mining.

- *DB* is the set of old transactions from the original database.
- *db* is the set of new incoming transactions (the increment).
- *DB+db* is the set of old and new incoming transactions (the resulting combined dataset).
- SCDB (X) is the support count of itemset X in DB.
- *SCdb* (X) is the support count of X in *db*.
- *SCDB + db* (X) is the support count of X in *DB+ db*.

Assume that the size of the increment *db* is less than the size of the original dataset *DB*. Define the support count (*SC*) of an itemset as the number of occurrences of that itemset in the dataset. Figure 6 is an illustration of the incremental approach in terms of the support count of *DB* (SC_{DB}) plotted against the support count of *db* (SC_{db}) (Imberman, Tansel, & Pacuit, 2004). For each point *(SC_{db}, SC_{DB})* in Figure 6, $SC_{db} + SC_{DB} = SC_{DB+db,}$ which is the support count of the new dataset. Let *minSCdb* be the minimum number of transactions to be frequent in *db* and *minSCDB* be the minimum number of transactions to be frequent in *DB*. Therefore, for all points G on line HC, *SCdb + SCDB = minSCDB+db. Line HC partitions the space of itemsets*. All itemsets above and including HC are frequent. All itemsets below HC are not frequent. Triangle HFG represents those itemsets that have become infrequent or have submerged. Triangle GIC represents those itemsets that have become frequent or emerged. The incremental discovery problem can be thought of as efficiently identifying those itemsets in triangles GIC and HFG.

Update With Early Pruning (UWEP; Ayan et al., 1999) has been shown to be an efficient incremental association rule algorithm. Like most incremental algorithms, UWEP uses information found in the increment to prune the search space. First UWEP scans *db* to find the counts for all the 1-itemsets. In the pruning step, the supersets of the itemsets in *DB* that are found not frequent in *DB + db* are pruned from *DB + db*. The frequent itemsets in DB, whose items are absent in *db*, are checked to see if $SC_{DB}(X) >= SC_{DB+db}(X)$. Frequent itemsets in *db* are looked at to see if they are frequent in *DB*. These are frequent by definition. Lastly, for all itemsets that are frequent in *DB* or *db* and have not yet been looked at, they are checked to see if they are frequent in *DB + db*.

Besides researching new ways for efficiently finding association rules, researchers have looked at the rules themselves, finding new types of association rules. Association rules are Boolean, looking only at the positive associations between items. Some researchers have looked at the negative dependencies as well, calling these rules

dependency rules (Silverstein, Brin, & Motwani 1998). A dependency rule can express concepts such as *males age 60 or greater = yes* and *smokes = no* ⇒ *buy beer = no*. Imberman et al. (2002) have shown that dependency rules are more expressive for medical data miming than association rules.

Since most data is not Boolean by nature, how can one show associations with numeric and categorical data? Quantitative association rules express associations such as *Age: 30 to 39* and *Owns car = yes* ⇒ *Median Income = 40,000*. One approach was to map each category in a categorical variable, to a Boolean variable and discretize the quantitative variables into intervals (Srikant & Agrawal, 1996). Each interval is mapped onto a Boolean variable. Then any Boolean association rule algorithm can be used to find rules. However, care needs to be exercised in partitioning each variable appropriately. If a quantitative variable is partitioned into too many smaller intervals, minimum support may not be found in any one interval. Therefore some well-supported rules may be missed. Also, confidence can decrease with larger intervals affecting the attainment of minimum confidence. Thus, small intervals might not get minimum support, while large intervals might not get minimum confidence. An approach to solve this problem is to consider all possible continuous ranges (Srikant & Agrawal, 1996). If we were to increase the interval size then we would have no more *minsup* problem. To take care of this we can combine adjacent intervals. But we may still have *minconf* problems. We can solve *minconf* problems by increasing the number of intervals. But doing both leads to two more problems. Given n intervals, there are on average $O(n^2)$ possible ranges. There is therefore a blow up in execution time. Given an interval with support, any range containing that interval also has support. This can lead to a blow up in the number of rules known as the many rules problem. Srikant and Agrawal (1996) posed a solution by setting a user-defined maximum on the size of the interval and using an interestingness measure to filter out uninteresting rules.

The 'many rules problem' has motivated research into the 'interestingness' of rules produced by association rule algorithms. Methods for interestingness involve ordering and grouping association rules in order to facilitate their use and interpretation. Metrics for ordering rules include measures such as confidence, added value, mutual information (Sahar & Mansour, 1999), and conviction measures (Brin, Motwani, Ullman, & Tsur, 1997). Objective interestingness measures seem to cluster into three groups when support and confidence levels are low. Interestingness measures in the same cluster produce similar rule orders. Sahar (1999) along with Mansour (1999) pruned the rule set by discarding 'uninteresting' rules. Sahar worked under the premise that simple rules would already be known by the user and can thus be pruned from the rule set. Sahar (2002) used clustering to group similar rules.

FUTURE TRENDS

Association rule discovery algorithms feed their results into organizational knowledge bases. An important issue is the maintenance and update of discovered association rules as new data becomes available. The incremental algorithms we have summarized above are very useful and cost effective for knowledge management. Research into the combination of sound knowledge management techniques and data mining techniques can make significant contributions to the business environment.

Research into the types of rules that can be generated using the techniques outlined in this article is ongoing. Reduced database scanning by improvements on the basic algorithm is another area of research activity. In addition much current research is being concentrated on finding better data structures for more efficient itemset processing (Gosta & Zhu, 2003). Association rule mining is a very active research field.

CONCLUSION

Association rule algorithms show co-occurrence of variables. One of the major problems inherent in Apriori, and algorithms like Apriori, is that there tends to be a large number of rules generated, some of which are commonly known. In addition, attempts to use the rules generated by association rule algorithms has met with mixed results. On the other hand, Apriori has also been shown to find less obvious patterns in the data (Cox, Eick, Wills, & Brachman, 1997), thereby discovering very valuable knowledge.

ACKNOWLEDGMENTS

This work was supported in part by a grant from The City University of New York PSC-CUNY Research Award Program, awards 65388-00-34 (Imberman) and 65406-00-34 (Tansel).

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD* (pp. 207-216).

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A.I. (1996). Fast discovery of association rules. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining* (pp. 307-328). Boston: AAAI/MIT Press.

Agrawal, R., & Srikant, R. (1994). *Fast algorithms for mining association rules.* IBM Res. Rep. RJ9839, IBM Almaden.

Aumann, Y., & Lindell, Y. (1999). A statistical theory for quantitative association rules. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 261 -270).

Ayan, N.F., Tansel, A.U., & Arkun, E. (1999). An efficient algorithm to update large itemsets with early pruning. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 287-291).

Brin, S., Motwani, R. Ullman, J., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. *Proceedings of ACM SIGMOD* (pp. 255-264).

Cox, K.C., Eick, S.G., Wills, G.J., & Brachman, R.J. (1997). Visual data mining: Recognizing telephone calling fraud. *Data Mining and Knowledge Discovery, 1*(2), 225-231.

Fayyad, U.M., Piaetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: An overview. *Advances in knowledge discovery and data mining.* Cambridge, MA: AAAI/MIT Press.

Fukada, T., Yasuhiko, M., Sinichi, M., & Tokuyama, T. (1996). Mining optimized association rules for numeric attributes. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 13-23).

Gosta, G., & Zhu, J. (2003). Efficiently using prefix-trees in mining frequent itemsets. *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations.* Retrieved from *http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-90/*

Imberman, S.P., Domanski, B., & Thompson, H.W. (2002). Using dependency/association rules to find indications for computerized tomography in a head trauma dataset. *Artificial Intelligence in Medicine, 26*(1), 55-68.

Imberman, S.P., Tansel, A.U., & Pacuit, E. (2004). An efficient method for finding emerging large itemsets. *Proceedings of the 3rd Workshop on Mining Temporal and Sequential Data, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Park, J.S., Chen, M.S., & Yu, P.S. (1995). An effective hash based algorithm for mining association rules. *Proceedings of ACM SIGMOD* (pp. 175-186).

Sahar, S. (1999). Interestingness via what is not interesting. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 332-336).

Sahar, S., & Mansour, Y. (1999). An empirical evaluation of objective interestingness criteria. *Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery* (pp. 63-74).

Sahar, S. (2002). Exploring interestingness through clustering: A framework. *Proceedings of the IEEE International Conference on Data Mining* (ICDM 2002) (pp. 677-681).

Savasere, A., Omiecinski, E., & Navathe, S. (1998). An efficient algorithm for mining association rules in large databases. *Proceedings of the IEEE International Conference of Data Engineering* (ICDE 1998) (pp. 494-502).

Silverstein, C., Brin, S., & Motwani, R. (1998). Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery, 2*(1), 39-68.

Srikant, R., & Agrawal, R. (1996). Mining quantitative association rules in large relational tables. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 1-12).

Toivonen, H. (1996). Sampling large databases for association rules. *Proceedings of the 22nd International Conference on Very Large Databases* (VLDB'96) (pp. 135-145).

KEY TERMS

Apriori: A level-wise algorithm for finding association rules. Apriori uses the support of an itemset to prune the search space of all itemsets. It then uses the confidence metric to find association rules.

Association Rule: Given a set $I = \{ i_1, i_2, i_3, \ldots i_n\}$ of items, any subset of I is called an itemset. Let X and Y be subsets of I such that $X \cap Y = \phi$. An association rule is a probabilistic implication $X \Rightarrow Y$.

Confidence: Given an association rule $X \Rightarrow Y$, the confidence of a rule is the number of transactions that satisfy $X \cup Y$ divided by the number of transactions that satisfy X.

Data Mining: One step of the KDD process. Can include various data analysis methods such as decision trees, clustering, statistical tests, neural networks, nearest neighbor algorithms, and association rules

Interestingness: Methods used to order and prune the set of rules produced by association rule algorithms.

This facilitates their use and interpretation by the user. Metrics for interestingness include measures such as confidence, added value, mutual information. and conviction measures.

Knowledge Discovery in Databases (KDD): A paradigm for the analysis of large datasets. The process is cyclic and iterative, with several steps including data preparation, analysis, and interpretation. KDD uses various methods from such diverse fields such as machine learning, artificial intelligence, pattern recognition, database management and design, statistics, expert systems, and data visualization.

Quantitative Association Rules: Shows associations with numeric and categorical data. Quantitative rules would express associations such as: *Age: 30 to 39* and *Owns car = yes -> Median Income = 40,000.*

Support: Given an association rule $X \Rightarrow Y$, the support of the rule is the number of transactions that satisfy or match $X \cup Y$, divided by the total number of transactions. Support is an indication of a rule's statistical significance.

UWEP: An incremental association rule algorithm. Incremental association rule algorithms use the information obtained in previous association rule processing to reduce the amount of dataset scans when new transactions are added.