ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 6, June 2017

Mining Approximate Functional Dependencies and Concept Similarities to Answer Imprecise Queries -A Review

[1] Jismol Abraham, [2]Dr. R Priya
[1] Research Scholar, Vels University,Chennai, India;
[2]Associate Professor,MCA Dept, Vels University,Chennai, India;
[1]jismol2007@gmail.com, [2]Priyaa.research@gmail.com

Abstract: ***Nowadays, the information technology has been growing rapidly with the evolution of larger databases and datasets. For viewing or retrieving information from larger databases is seems to be a challenging task. The discovery of functional dependencies in a dataset is of great importance for database redesign, anomaly detection and data cleansing applications. Objective: The main objective of this paper is to fetch the common similar functional details from the database and splits them separately and provide different set of instructions for those separated information. Finding: In this paper, a new algorithm is designed called SFD (Similarity Functional Dependancy) algorithm for discovering all functional dependencies in a dataset. SFD follows a depth first traverse of the attribute lattice that combines aggressive pruning and efficient result verification. Novelty: This paper describes the concept about the fetching of data from the database in which there is repeated information or same details i.e. database with similar functional values. Improvement: This concept has been implemented with help of the similarity functional dependencies algorithm, which helps us to find out the repeated or similar database values or information. The new approach is able to scale far beyond the already existing algorithm known as functional dependency.***

Keywords: ***Data Mining, Hierarchical algorithm, SFD Algorithm, Anomaly Detection, Prediction.***

I. INTRODUCTION

Data Mining**:** Data mining is the process of applying these methods to data with the intention of uncovering hidden patterns. Data mining or data mining technology has been used for many years various fields such as business, research workers and governments. It is used to sift through volumes of data such as airline passenger trip information, population data and marketing data to generate market research reports, although that reporting is sometimes not considered to be data mining. Figure1 depicts how the end-user wants to view or retrieve any data from the database**.** For retrieving the larger dataset value, the user can generate an input value to access that file. In the next step the user input value pattern is evaluated for the task of preprocessing. After data cleansing, integration and selection, the data warehouse responds with the required value for user inputs. And then the data mining engine evaluates the pattern value and senses the data warehouse containing the entire dataset value.

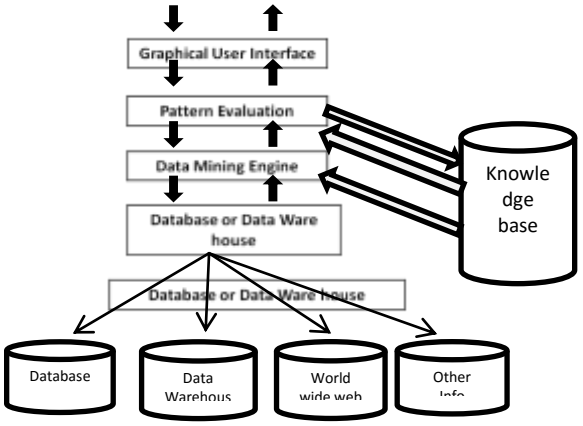

Figure1. Pattern value evaluation in data mining engine

Data mining commonly involves four classes of tasks:
(1) classification, arranges the data into predefined groups; (2) clustering, is like classification but the groups are not predefined, so the algorithm will try to group similar items together; (3) regression, attempting to find a function which models the data with the least error; and (4) association rule learning, searching for relationships between variables. According to Han and Kamber, data mining functionalities include data characterization, data discrimination, association analysis, classification, clustering, outlier analysis, and data evolution analysis. Data characterization is a summarization of the general characteristics or features of a target class of data. Data discrimination is a comparison of the general features of target class objects with the general features of objects from one or a set of contrasting classes. Association analysis is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. Classification is the process of finding a set of models or functions that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. Clustering analyzes data objects without consulting a known class model. Outlier and data evolution analysis describe and model regularities or trends for objects whose behaviour changes over time.

II. RESEARCH METHODS

2.1 Hierarchical Algorithm:
The hierarchical algorithm is an efficient and normal algorithm. This algorithm works like a process of FCFS (First Come First Serve). The data which are entered to the database will be stored in a sequential order. This method has been used frequently for the easy way of adding data or information to the database. In a hierarchal method, separate clusters are finally joined into one cluster. The density of the data points is employed to determine the relevant clusters. The main advantage is that, it uses less computation costs in term of combinatorial number of data points. After performing the merging or splitting process, it is very rigid and unable to reverse it. So any decisions prior to the earlier mistakes can't be rectified. Generally, hierarchical clustering algorithms can be divided into two categories: Divisive and Agglomerative. Agglomerative clustering performs the bottom-up strategy, in which it initially considers each data point as a singleton cluster. After that, it continues by merging all those clusters until all points are combined into a single cluster. A tree graph is used to represent the output. Then the algorithm splits back the single cluster in gradual manner until the required number of clusters are obtained. To be more specific, two major steps are involved. First step, is to choose a suitable number of clusters to split. Second is to determine the best approach on how to split the selected clusters into two new clusters. This method creates a hierarchical decomposition of the given set of data objects. It can classify hierarchical methods on the basis of how the hierarchical decomposition is formed. 2.2 Differential Dependencies:
Differential Dependencies specifies constraints on difference, called differential functions, instead of identification functions in traditional dependency notations like functional dependencies. Informally, a differential dependency states that if two tuples have distances on attributes X agreeing with a certain differential function, then their distances on attributes Y should also agree with the corresponding differential function on Y. For example, [date (≤ 7)] $\rightarrow$ [price (< 100)] states that the price difference of any two days within a week length should be no greater than 100 dollars. Such differential dependencies are useful in various applications, for example, violation detection, data partition, query optimization, record linkage, etc. Data dependencies, such as Functional Dependencies (FDs), are traditionally used for schema design, integrity constraints, and query optimization and so on, with respect to schema quality in databases. Conventional dependencies, originally proposed for schema-oriented issues, are defined based on equality function, that is, attribute values are compared according to equality.

2.3 SFD Algorithm:
Similarity Search and Data Mining have become widespread problems of modern database applications involving complex objects such as Multimedia, CAD, Molecular Biology, Sequence Analysis, etc. A common approach to grasp the intuitive idea of similarity by a formal means is to translate complex objects into multidimensional vectors by a feature transformation which allows retrieval of the most similar objects to a given query object (similarity search) but also to analyze the complete set of complex objects with respect to clusters, outliers, correlations etc. (data mining). According to SFD algorithm several areas of applications where the classical feature approach is not sufficient. Example applications include Biometric Identification, Medical Imaging, Electronic Commerce and Share Price Analysis. This shows that existing feature based similarity models fail due to different reasons, e.g. because they do not cope with the uncertainty which is inherent to their

feature vectors (biometric identification) or because they do not integrate application specific methods into the similarity model (share price analysis, medical imaging). It surveys the challenges and possible solutions to these problems to direct future research. If the structure of the information to be searched is sufficiently simple, such as in one-dimensional numerical attributes or character strings, search problems can be considered as solved. Database management systems (DBMS) provide index structures for the management of such data [BM 77, Com 79] which are well-understood and widely applied. Requirements of traditional applications such as accounting and billing are perfectly met by a commercial DBMS. Therefore, the information infrastructure of most enterprises is based on products such as Oracle or Informix. Recently, an increasing number of applications has emerged which process large amounts of complex, application specific data objects.

2.4 Anomaly Detection:

Anomaly detection (also outlier detection) is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset. Typically the anomalous items will translate to some kind of problem such as bank fraud a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions. In particular, in the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object. Many outlier detection methods particularly unsupervised methods will fail on abuse and network intrusion detection, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro clusters formed by these patterns. Anomaly detection's diagrammatic representation is shown in figure 2.

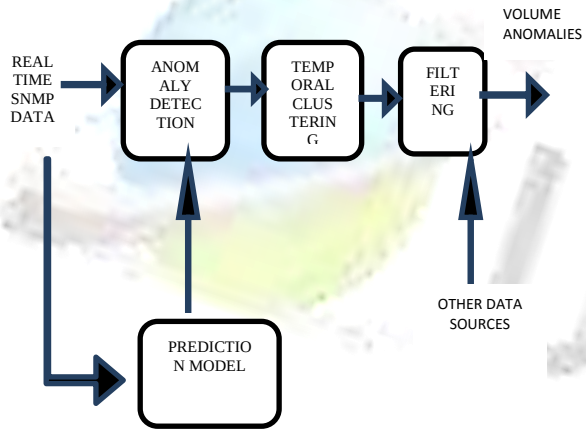

Figure2. Architecture of anomaly detection

Three broad categories of anomaly detection techniques exist. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly

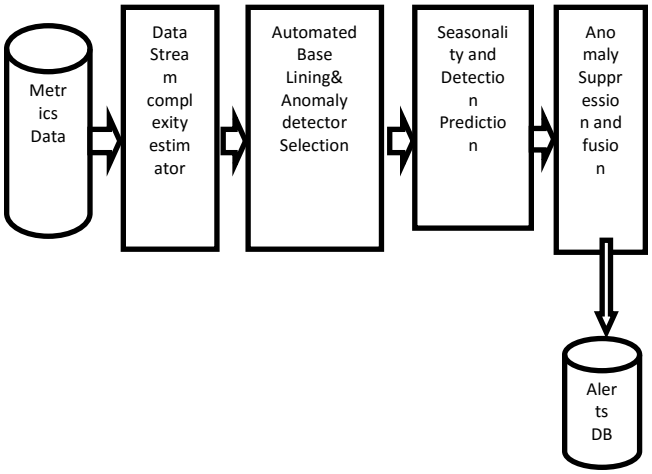

Figure3. Data stream estimator search

III. INCREASING THE EFFICIENCY OF SFD ALGORITHMS

Database dependencies, such as functional and multivalue dependencies, express the presence of structure in database relations that can be utilized in[1] the database design process. The discovery of database dependencies can be viewed as an induction problem, in which general rules (dependencies) are obtained from detection techniques require a data set that has been labeled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then testing the likelihood of a test instance to be generated by the learnt model.

Application of Anomaly Detection:

Anomaly detection is applicable in a variety of domains, such as intrusion detection, fraud detection, fault detection, system health monitoring, event detection in sensor networks, and detecting Eco-system disturbances. It is often used in preprocessing to remove anomalous data from the dataset. In supervised learning, removing the anomalous data from the dataset often results in a statistically significant increase in accuracy. Figure 3 shows the application of anomaly detection specific facts. This viewpoint has the advantage of abstracting away as much as possible from the particulars of the dependencies. The algorithms in this paper are designed such that they can easily be generalized to other kinds of dependencies. Differential Dependencies: Reasoning and Discovery investigated a number of aspects of DDs, the most important of which, since they form the basis for the other topics investigated, exists the consistency problem (determining whether there exists a relation instance that satisfies a set of DDs) and the implication problem (determining whether a set of DDs logically implies another DD). Concerning these problems, a number of results were claimed in “Differential Dependencies: Reasoning and Discovery”. In this article [2] a detailed analysis of the correctness of these results. The outcomes of proposed analysis are that, for almost every claimed result, there are either fundamental errors in the proof or the result is false. For some of the claimed results the corrected proofs can be given, but for other results their correctness remains open. Differential dependencies (DDs) capture the relationships between data columns of relations in 3. They are more general than functional dependencies (FDs) and the difference is that DDs are defined on the distances between values of two tuples, not directly on the values.. This paper shows some properties of DDs and conducts a comprehensive analysis on how sampling affects the DDs discovered from data. Most advanced methods for automating the design of the data warehouse in4 carry out this process from relational OLTP systems, assuming that a RDBMS is the most common kind of data source may find, and taking as starting point a relational schema.

In contrast, this paper proposes to rely instead on a conceptual representation of the domain of interest formalized through a domain ontology expressed in the DL-Lite Description Logic. In the proposed approach, an algorithm to discover functional dependencies from the domain ontology that exploits the inference capabilities of DL-Lite, thus fully taking into account the semantics of the domain. This also provides an evaluation of suggested approach in a real-world scenario.

Association rules are interesting correlations among attributes in a database5. These rules have many applications in areas ranging from e-commerce to sports to census analysis to medical diagnosis. The discovery of association rules is an extremely computationally expensive task and it is therefore imperative to have fast scalable algorithms for mining these rules. In this thesis, an efficient techniques for discovering association rules from large databases and for removing redundancy from these rules so as to improve the quality of output. This also handles growing databases. It utilizes previous mining results to efficiently mine the current database after it has been updated with fresh data. It also handles situations where the mining specifications over the current database differ from those used over the original database, a common occurrence in practice. The concept of matching dependencies (MDs) has recently been proposed for specifying matching rules for object

identification. Similar to the functional dependencies (with conditions), MDs can also be applied to various data quality applications such as detecting the violations of integrity constraints.This paper includes the study regarding the problem of discovering similarity constraints for matching dependencies from a given database instance. First, this introduces the measures, support and confidence, for evaluating the utility of MDs in the given data. Then analysing the discovery of MDs with certain utility requirements of support and confidence. Exact algorithms are developed, together with pruning strategies to improve the time performance. Since the exact algorithm has to traverse all the data during the solution6 which only uses part of the data. A bound of relative errors introduced by the approximation is also developed. Finally, experimental evaluation demonstrates the efficiency of the proposed methods.computation, this paper proposes an approximate

IV. RECENT ADVANCES IN SFD ALGORITHM:

As the nature of the problem is exponential in the number of attributes none of the existing approaches can be applied on large datasets. This paper introduces new algorithm SFD7 for discovering all functional dependencies in a dataset following a depth first traversal strategy of the attribute lattice that combines aggressive pruning and efficient result verification. The new approach is able to scale far beyond existing algorithms for up to 7.5 million tuples, and is up to three orders of magnitude faster than existing approaches on smaller datasets. This8 work presents an overview of additional details where the original papers were ambiguous or incomplete. Evaluation of careful re-implementations of all algorithms spans a broad test space including synthetic and real-world data. This paper shows that all functional dependency algorithms optimize for certain data characteristics and provide hints on when to choose which algorithm. In summary, however, all current approaches scale surprisingly poorly, showing potential for future research.

V. CONCLUSION AND FUTURE WORK

The individual strengths and weakness of the newly designed SFD [Similarity Functional Dependancy] algorithm has been discussed in this paper. The evaluation constitutes the first qualitative comparison of all FD-discovery algorithms and analyzes both execution times and memory consumptions. With the extrapolation of evaluation results, this paper provides the reader with a tool to choose the fastest algorithm for any given dataset. In summary, experiments have shown that FD discovery is still an open research. None of the state-of- the art algorithms in the experiments scales to datasets with hundreds of columns or millions of rows. Given a dataset with 100 columns and 1 million rows, which are a reasonable size for a table, SFD, Depth Miner, Fast FDs will starve in runtime, whereas Tame, Fun, and FD Mine will use up any available memory. This observation indicates potential for future research. For a user to make sense of such large result sets, further techniques are needed, such as ranking FDs by interestingness or visualizing attributes and their FDs. This interpretation of dependencies is a further important topic for future research for all data profiling approaches.

References

[1] P. A. Flach and I. Savnik. Database dependency discovery: A machine learning approach. AI Commun., 12(3):139--160, 1999.
[2] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. ACM Transactions on Database Systems, 36:16, 2016.
[3] R. King and J. Oil. Discovery of functional and approximate functional dependencies in relational databases. J. Applied Math and Decision Sciences, 7(1):49--59, 2003.
[4] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann. Discovering Functional Dependencies for
Multidimensional Design. Proc. VLDB Endow. 8(10):1082--1093, 2015.
[5] S. Kwashie, J. Liu, J. Li, and F. Ye. Efficient discovery of differential
dependencies through association rules mining. In Proceedings of Australasian Database Conference, ADC '15, pages 3--15, 2015.
[6] S. Song and L. Chen. Efficient discovery of similarity constraints for matching dependencies. Data & Knowledge Engineering, 87:146--166, 2016.
[7] Z.Abedjan, P.Schulze, and F. Naumann. Efficient functional dependency discovery. In Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, CIKM '14, pages 949--958, 2014.
[8]T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, and F. Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. Proc. VLDB Endow. 8(10):1082--1093, 2015.
[9] D.Muruga Radha Devi and P.Thambidurai.Similarity Measurement in recent biased time series databases using different clustering methods, IJST Vol 7(2),189- 198,February 2014.
[10] Debajit Sensarma and Samr Sen Sarma-Survey On Different graph Based Anomaly Datection Techniques. IJST Vol8(51), doi;10.17485.November 2015.
[11] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma, “Dynamic constraints for record matching,” VLDB J., vol. 20, pp. 495–520, 2011.
[12] J. Liu, J. Li, C. Liu, and Y. Chen, “Discover dependencies from data—A review,” IEEE Trans. Knowl. Data Eng., vol. 24, no. 2, pp. 251–264, Feb. 2012.

[13] M. _I. S€ozat and A. Yazici, "A complete axiomatization for fuzzy functional and multivalue dependencies in fuzzy database relations," Fuzzy Sets Syst., vol. 117, no. 2, pp. 161–181, 2001.
[14] M. W. Vincent, J. Liu, and C. Liu, "Strong functional dependencies and their application to normal forms in XML," ACM Trans. Database Syst., vol. 29, no. 3, pp. 445–462, 2004.
[15] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," IEEE Trans. Knowl. Data Eng., vol. 19, no. 1, pp. 1–16, Jan. 2007.
[16] D. A. Simovici, D. Cristofor, and L. Cristofor, "Impurity measures in databases," Acta Informatica, vol. 38, no. 5, pp. 307–324, 2002
[17] I. F. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga, "CORDS: Automatic discovery of correlations and soft functional dependencies," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2004, pp. 647–658.
[18] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. B. Zdonik, "Correlation maps: A compressed access method for exploiting soft functional dependencies," Proc. VLDB Endowment, vol. 2, pp. 1222–1233, 2009.
[19] D. Z. Wang, X. L. Dong, A. D. Sarma, M. J. Franklin, and A. Y. Halevy, "Functional dependency generation and applications in pay-as-you-go data integration systems," in Proc. 7th Int. Workshop Web Databases, 2009, pp. 1–6.
[20] W. Chen, W. Fan, and S. Ma, "Incorporating cardinality constraints and synonym rules into conditional functional dependencies," Inf. Process. Lett. vol. 109, no. 14, pp. 783–789, 2009.
[21] G. Cormode, L. Golab, K. Flip, A. McGregor, D. Srivastava, and X. Zhang, "Estimating the confidence of conditional functional dependencies," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2009, pp. 469–482.
[22] L. Bravo, W. Fan, F. Geerts, and S. Ma, "Increasing the expressivity of conditional functional dependencies without extra complexity," in Proc. Int. Conf. Data Eng., 2008, pp. 516–525.
[23] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies," Proc. VLDB Endowment, vol. 1, no. 1, pp. 376–390, 2008.
[24] W. Chen, W. Fan, and S. Ma, "Analyses and validation of conditional dependencies with built-in predicates," in Proc. 20th Int. Conf. Database Expert Syst. Appl., 2009, vol. 5690, pp. 576–591.
[25] L. T. H. Vo, J. Cao, and W. Rahayu, "Discovering conditional functional dependencies in XML data," in Proc. Australasian Database Conf., 2011, pp. 143–152.
[26] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian, "Metric functional dependencies," in Proc. Int. Conf. Data Eng., 2009, pp. 1275–1278.
[27] J. Baixeries, M. Kaytoue, and A. Napoli, "Computing similarity dependencies with pattern structures," in Proc. Int. Conf. Concept Lattices Their Appl., 2013, pp. 33–44.
[28] J. Baixeries, "Computing functional dependencies with pattern structures," in Proc. Int. Conf. Concept Lattices Appl., 2012, pp. 175–186.
[29] L. Bertossi, S. Kolahi, and L. Lakshmanan, "Data cleaning and query answering with matching dependencies and matching functions," Theory Comput. Syst., vol. 52, no. 3, pp. 441–482, 2013.
[30] Z. Bia and M. Shan, "Review of data dependencies in data repair," J. Inf. Comput. Sci., vol. 9, no. 15, pp. 4623–4630, 2012.