Distributed Higher Order Text Mining

Shenzhi LI, Christopher D. JANNECK, Xiaoning YANG, Aditya BELAPURKAR, Osama A. KHAN, Tianhao WU, Murat GANIZ, Mark D. DILSIZIAN and William M. POTTENGER, *Member, IEEE*

***Abstract*--The burgeoning amount of textual data in distributed sources combined with the obstacles involved in creating and maintaining central repositories motivates the need for effective distributed information extraction and mining techniques. Recently, as the need to mine patterns across distributed databases has grown, Distributed Association Rule Mining (D-ARM) algorithms have been developed. These algorithms, however, assume that the databases are either horizontally or vertically distributed. In the special case of databases populated from information extracted from textual data, existing D-ARM algorithms cannot discover rules based on higher-order associations between items in distributed textual documents that are neither vertically nor horizontally distributed, but rather a hybrid of the two. In this article we present D-HOTM, a framework and system for Distributed Higher Order Text Mining. Unlike existing algorithms, those encapsulated in D-HOTM require neither full knowledge of the global schema nor that the distribution of data be horizontal or vertical. D-HOTM discovers rules based on higher-order associations between distributed database records containing the extracted entities. A theoretical framework for reasoning about record linkage is provided to support the discovery of higher-order associations. In order to handle record linkage, the traditional evaluation metrics employed in ARM are extended. The implementation of D-HOTM is based on the TMI and tested on a cluster at the National Center for Supercomputing Applications (NCSA). A sample manual run, results of experimental runs on the NCSA clusters, and theoretical comparisons demonstrate the performance and relevance of D-HOTM in e-marketplaces, law enforcement and homeland defense.**

***Index Terms*—Algorithms, Performance, Experimentation, Theory**

I. INTRODUCTION

Association Rule Mining (ARM) is one of the most widely used algorithms in data mining. Generating rules based on statistics of item co-occurrence, ARM produces output in the form of propositional rules. Co-occurrence refers to instances where two or more items appear in the same context, and is also called 1st-order association. Much work has been done developing techniques for generating, analyzing and measuring 1st-order associations [1]-[6], but most techniques do not support mining across transaction boundaries. Notable exceptions include sequence mining and multi-relational ARM. These are examples of approaches that discover *higher-order* associations.

Manuscript submitted May 10, 2006 – under an approved extension from the original May 1, 2006 deadline. This work was supported in part by NSF grant number 0534276 and NIJ grant numbers 2003-IJ-CX-K003, 2005-93045-PA-IJ and 2005-93046-PA-IJ.

Higher-order associations are formed by linking different contexts (e.g., transactions) through one or more common items. Consider the following example from traditional market-basket analysis: if customer "A" purchases {computer, OS}, customer "B" buys {laptop, OS} and customer "C" gets {laptop, mouse, battery}, then several higher-order associations can be formed. These include computer-to-laptop through OS, OS-to-mouse through laptop, computer-to-battery through OS and laptop, etc. The first two associations are termed 2nd-order since they each span two contexts, while the computer-to-battery association is 3rd-order. Overall, any association greater than 1st-order (i.e., spanning at least two contexts) is termed a higher-order association.

Higher-order associations are currently employed in a number of real world applications including medical research, marketing analysis, law enforcement and homeland defense. In the field of medicine, for example, Literature Based Discovery [7] has been used to uncover a higher-order association between Fish Oil and Raynaud’s disease in medical literature, leading to a potential new treatment [8]. An important law enforcement application that employs 2nd-order associations is COPLINK® Detect [9], which assists law enforcement personnel through textual entity extraction and link-generation through the use of a semantic network. Mooney et al. [10] discusses a link discovery algorithm, as part of the DARPA Evidence Extraction and Link Discovery program that uses Inductive Logic Programming (ILP) in its mining of multi-relational data.

Several research efforts are also revealing promising results on the utility of higher-order associations. Latent Semantic Indexing (LSI) is a time-intensive yet powerful document indexing technique that can reveal hidden or latent relationships between terms. In [1], Kontostathis and Pottenger present experimental evidence suggesting a strong correlation between 2nd-order association of terms and the performance of LSI in terms of F_β, the harmonic mean of precision and recall. The authors confirm this empirical evidence through a mathematical proof that LSI is implicitly dependent upon 2nd- and higher-order associations. In related work, based on statistical comparisons of distributions of higher-order association frequencies, Ganiz and Pottenger report that classes of instances in labeled training data may be separable based on the characteristics of the higher-order associations alone [11].

As noted, existing work in ARM also demonstrates the utility of higher-order associations. In the field of sequence mining, an example is sequential pattern mining algorithms that return all frequent subsequences in a sequence database [12], [13]. This and subsequent work allows the discovery of transactions that reference the same object through the use of 2nd-order associations. In a related effort, Tan et al. [14], [15] present an Apriori-based algorithm that discovers 2nd-order associations between transactions, and use it to portray an association between the men's jewelry page and other jewelry pages from Web server logs. In a survey of this field, Spiliopoulou et al. [16] suggest that there are many possible applications for higher-order association (or link analysis) research, including models for ensemble learning and other time-dependent phenomena that can only be appropriately explained through such associations. A final example from the field of distributed ARM is Li et al. [17], which provides a theoretical framework, algorithm and initial empirical results demonstrating the extension of the basic ARM framework to the distributed higher-order ARM domain.

In this article we present D-HOTM, a framework and system for Distributed Higher Order Text Mining. Unlike existing algorithms, those encapsulated in D-HOTM require neither full knowledge of the global schema nor that the distribution of data be horizontal or vertical. D-HOTM discovers rules based on higher-order associations between distributed database records containing extracted entities. A theoretical framework for reasoning about record linkage is provided to support the discovery of higher-order associations. In order to handle record linkage, the traditional evaluation metrics employed in ARM are extended. The implementation of D-HOTM is based on the TMI and tested on a cluster at the National Center for Supercomputing Applications (NCSA). A sample manual run, results of experimental runs on the NCSA clusters, and theoretical comparisons demonstrate the performance and relevance of D-HOTM in e-marketplaces, law enforcement and homeland defense.

II. RELATED WORK

Association rule mining (ARM) discovers associations between items [1], [2]. Let I be a set of distinct items, or attribute-value pairs. Let D be a set of records (or transactions), where each record has a unique identifier and contains a set of items. A set of items is called an itemset. An itemset with k items is called a k-itemset. The support for an itemset is the number of transactions in D for which the itemset is a subset. The itemsets meeting the user defined minimum support are referred as frequent itemsets. Given two distinct frequent itemsets X and Y, association rules take the form "$X \Rightarrow Y$". The confidence of a rule is defined as the support of XY divided by the support of X. As noted in the Introduction, most existing ARM algorithms identify only 1^{st}-order associations, i.e., co-occurrence in the same context. On the other hand, higher-order association occurs between different contexts, linking contexts through items such as the value of an attribute in a database. There are three types of existing ARM algorithms that identify higher-order associations: multi-relational ARM, sequential pattern mining, and indirect association. In what follows we deal with each of these in turn.

A. Multi-Relational ARM

Multi-relational ARM is a type of ARM algorithm designed specifically to mine rules across tables in a single database [18], [19]. In fact, multi-relational data mining in general (not limited to ARM) is an emerging research area that enables the analysis of complex, structured types of data such as sequences in genome analysis. Similarly, there is a wealth of recent work concerned with enhancing existing data mining approaches to employ relational logic. WARMR, for example, is a multi-relational enhancement of Apriori presented by Dehaspe and Raedt [19].

WARMR [19] is an Apriori-based algorithm. The key difference between WARMR and Apriori is that WARMR uses sets of atoms based on first order logic as an extension to the itemsets used in Apriori. The atoms are also referred to as queries if all the variables are quantified and the set is ordered. An example query is: Key(X) $\leftarrow$ Buys(X,Y,card), Property(X,loyal). Here X is a variable that takes unique customer IDs as its range of values. Buys and Property are predicates, and can be thought of as different tables in a relational database. The Buys predicate records purchases – e.g., customer value(X) buys some item value(Y) using a particular credit card. The Property predicate specifies that customer value(X) has the property of being loyal. The key value(X) connects the two predicates. In this example, it is actually only the body of the Horn clause that is termed a query. The support of the query is defined as the number of variable bindings for the head of the Horn clause Key(X).

Using atom sets in multi-relational ARM leads to a change in the relation used for counting and evaluation. On the one hand, Apriori makes use of the subset relation in itemset counting and evaluation. In multi-relational ARM, however, the subset relation cannot be used to express the relations among atom sets. Instead, WARMR uses θ-subsumption. An atom set C subsumes an atom set D, denoted as C $\geq$ D, if there is a substitution θ such that C $\theta \subseteq$ D. The θ-subsumption relation induces an equivalence relation ~, which is defined as follows: C ~ D iff C $\geq$ D and D $\geq$ C. For example, given the two queries Q1: Buys(X, Y), Likes(X, cola) and Q2: Buys(Z, Z), Likes(Z, cola), Q2 subsumes Q1 with $\theta = \{X/Z, Y/Z\}$, while Q1 does not subsume Q2.

Although WARMR provides a sound theoretical basis for multi-relational ARM, it does not seriously address the efficiency of computation. In fact, the runtime performance of WARMR depends heavily on the implementation of θ-subsumption, and because θ-subsumption is NP-complete, performance is poor.

In addition, aside from these multi-relational approaches, ARM algorithms are generally based on propositional logic. Practically speaking, propositional rules are directly applicable due to the perspicuity of a propositional representation. Rules generated using a multi-relational ARM algorithm, however, must first be instantiated. As a result, propositional rules are widely employed in a variety of association rule mining applications. In summary, although multi-relational ARM has extended previous ARM algorithms to discover higher-order relations, it suffers from high complexity and a non-propositional representation that prevents scaling to real-world data sets.

B. Sequence Mining

Sequential pattern mining is a data mining approach that discovers frequent subsequences as patterns in a sequence database. Almost all of the proposed methods for mining sequential patterns and other time-related frequent patterns are Apriori-based; i.e., based on the Apriori property proposed in association rule mining [1], [20] that states that any super-pattern of a non-frequent pattern cannot be frequent.

Sequential pattern mining was introduced by Agrawal and others in [12], [13]. Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified minimum support threshold, sequential pattern mining discovers all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than the minimum support. For example, let the element be one transaction made in supermarket, then one sequence will contain all the transactions made by the same customer, e.g., {customer_ID, (a, b), (d, e), … }. In this case (a, d) is an example of a subsequence.

In later work, Mannila et al. introduce an efficient solution to the discovery of frequent patterns in a sequence database [21]. Chan et al. [22] study the use of wavelets in time-series matching and Faloutsos et al. [23] and Keogh et al. [24] propose indexing methods for fast sequence matching using R* trees, the Discrete Fourier Transform and the Discrete Wavelet Transform. Toroslu et al. introduce the problem of mining cyclically repeated patterns [25]. Han et al. introduce the concept of partial periodic patterns and propose a data structure called the Max Subpattern Tree for finding partial periodic patterns in a time series [26]. To accommodate the phenomenon that the system behavior may change over time, a flexible model of asynchronous periodic patterns is proposed in [27]. In [28], instead of frequently occurring periodic patterns, statistically significant patterns are mined. Aref et al. extend Han’s work by introducing algorithms for incremental, online and merge mining of partial periodic patterns [29]. Bettini et al. propose an algorithm to discover temporal patterns in time sequences [30].

In summary, sequential pattern mining identifies higher-order associations by linking transactions that reference a common object such as a customer. Although it employs a propositional framework for rules, the higher-order associations mined (i.e., sequential patterns) are limited to 2nd-order.

C. LBD Using ARM

Literature Based Discovery (LBD) is a technique that aims to identify indirect relationships between concepts in disjoint science literatures [7]. Based originally on a manual technique, LBD now generally refers to the study of more efficiently, automatically or correctly determining the intermediate concepts (B) that link a start concept (A) to a target concept (C) in such a way that $A \rightarrow B \rightarrow C$. One prominent discovery made through this process is that the concept of "Fish Oil" (and related literature discussing this) can be linked to the concept "Raynaud's disease" (and accompanying literature) through the concepts of "blood viscosity" and "blood cell rigidity" (which are higher in Raynaud's disease patients and lowered by treatment with fish oil). Since none of the literature on fish oil mentioned Raynaud's disease (and vice versa), after undergoing clinical testing this 2nd-order association was considered a novel discovery.

Pratt and Yetisgen-Yildiz [31] present an LBD system, LitLinker, which incorporates association rule mining to find correlations between concepts. Beginning with a given start concept (A), any correlated concepts found in this step act as the linking concepts. The same process is repeated using each linking concept for separate literature searches. The resulting correlated concepts for each linking concept create the total set of target concepts. However, LitLinker only uses ARM to discover 2-itemsets, and the discovered associations are limited to 2nd-order only.

D. Indirect Association

More recently, methods that address 2nd-order associations have been discussed in the context of the e-Marketplace. Tan et al. [14] propose a method of discovering indirect (2nd-order) associations between items in a database of transactions. The proposed INDIRECT algorithm is introduced in [14] and uses Apriori to link two items that are both highly dependent on a mediator set. The algorithm consists of two major phases: (1) extract all frequent itemsets using Apriori and (2) discover all indirect associations by candidate generation and pruning. For example, two frequent itemsets, {*a, b, c, d*} and {*a, b, d, e*}, can be joined together to produce a candidate indirect association, <*c, e,* {*a, b, d*}>. In this example the candidate association between *c* and *e* satisfies the mediator support condition because it is created by joining a frequent itemset that occurs in multiple records. This approach has similarities with sequence mining; in effect, the mediator set replaces the item held in common that is used to create the sequence (e.g., customer_ID).

Tan et al. follow up this work by evaluating the utility of the INDIRECT algorithm in [21]. The dataset consisted of Web server logs from an e-commerce organization. The authors successfully derived an association between the men’s jewelry page and other jewelry pages (e.g., anniversary page, ladies page) through the mediator "jewelry." This suggested that users looking for male jewelry are rarely interested in other jewelry. In summary, although [21] discovers higher-order associations, like sequence mining it does not extend beyond 2nd-order.

Also in this field, Spiliopoulou et al. perform a survey of research in higher-order association mining [16]. The authors suggest that there is significant potential for the application of higher-order association including: (1) facilitation of combining data mining strategies and (2) modeling many every-day phenomena, particularly over time, which require higher-order explanations and (3) a higher-order algorithm could be used to monitor and describe changes in rule sets.

E. Higher Order Co-Occurrence

In our previous work in [1], we prove mathematically that Latent Semantic Indexing (LSI), a well-known approach to information retrieval, implicitly depends on higher-order co-occurrences. We also demonstrate empirically that higher-order co-occurrences play a key role in the effectiveness of systems based on LSI. LSI can reveal hidden or latent relationships among terms, as terms semantically similar lie closer to each other in the LSI vector space. This can be demonstrated using the LSI term-term co-occurrence matrix. Assume a simple document collection where D1 is {human, interface} and D2 is {interface, user}. Clearly the terms "human" and "user" do not co-occur in the co-occurrence matrix of this simple two-document collection. After applying LSI, however, the reduced representation co-occurrence matrix may have a non-zero entry for "human" and "user" thus implying a similarity between the two terms. This is an example of second-order co-occurrence; in other words, there is a second-order path between "human" and "user" through "interface." In a related effort in [32], Edmonds uses higher order co-occurrence to solve a component of the problem of lexical choice, which identifies synonyms in a given context. In another effort, Zhang et al. [33] use second-order co-occurrence to improve the runtime performance of LSI. Second- and higher-order co-occurrence has also been used in a number of other applications including word sense disambiguation [34] and in a stemming algorithm [35].

F. Object Isomerism

In the D-HOTM system we employ techniques to link records based on identifying the same entities in multiple (distributed) records. This requires matching techniques known as object isomerism. Among other techniques developed for object isomerism, Fellegi and Sunter [36] employed string comparisons, and provided a formal mathematical model for estimating optimal parameters (i.e., probabilities of matching records) using the likelihood ratio. Jaro [37] introduces a string comparator measure that gives values of partial agreement between two strings. The string comparator accounts for length of strings and partially accounts for the types of errors typically made in alphanumeric strings by human beings. Tucker [38] uses Hamming Distance to measure the number of positions with different characters between two words of the same length combined with a shift, or algorithm to compute an array with ones and zeros that maps matches and mismatches in letters. Many other approaches have been taken, including several not based on string comparators such as a method cited in Winkler [39], that employs a neural network to separate records that match from those that do not. This approach has the benefit of not requiring a conversion process of characters to numerical values that have a well-defined meaning of true distance. Canopy Clustering with TFIDF (Term Frequency/Inverse Document Frequency) [40], [41] forms blocks of records based on the subset of records placed in the same canopy cluster. A canopy cluster is formed by choosing a record at random from a candidate set of records (initially, all records) and then populating its cluster with records within a certain threshold distance. The record chosen at random and any records within a certain threshold distance are then removed from the candidate set of records.

This related work demonstrates the efforts and results driving further research into higher-order links. Across many of these sub-fields, this research encounters the challenges of poor practical performance and limitations to 2nd order links. Encouraged by the potential demonstrated for the use of 2nd and higher order links, the next section discusses the architecture of the Distributed Higher Order Text Mining (D-HOTM) System – a link analytics system which provides solutions for many of the challenges in this field.

III. D-HOTM Architecture

The D-HOTM System can be conceptually divided into three subsystems: Mining, Controls, and Post-Processing. The Mining subsystem incorporates all aspects of the computations relating to data preparation, processing, transformations, mining, linking and other computations. The Controls subsystem provides an interface to specify the parameters of all actions performed by the Mining subsystem, and the Post-Processing subsystem provides tools and mechanisms to assist in the analysis and exploration of output generated by the Mining subsystem. Fig. 1 provides a conceptual architectural overview of the system.

The Mining subsystem forms the core of the D-HOTM system. One of the components of this subsystem is Entity Extraction and XML Parsing. This component is used to generate input for mining by extracting entities from raw documents and writing them into an XML format. An XML parser then reads these files and forms records used in the association rule mining process. The Data Distributor component is responsible for exchange of Global Identifiers and sharable records among the multiple D-HOTM sites working in a distributed environment. Other computations in this subsystem are discovering and merging linkable records and generating itemsets or association rules that span multiple records. This functionality is achieved through different components of the subsystem, working together. Each of these components will be discussed in more detail in the following sections.

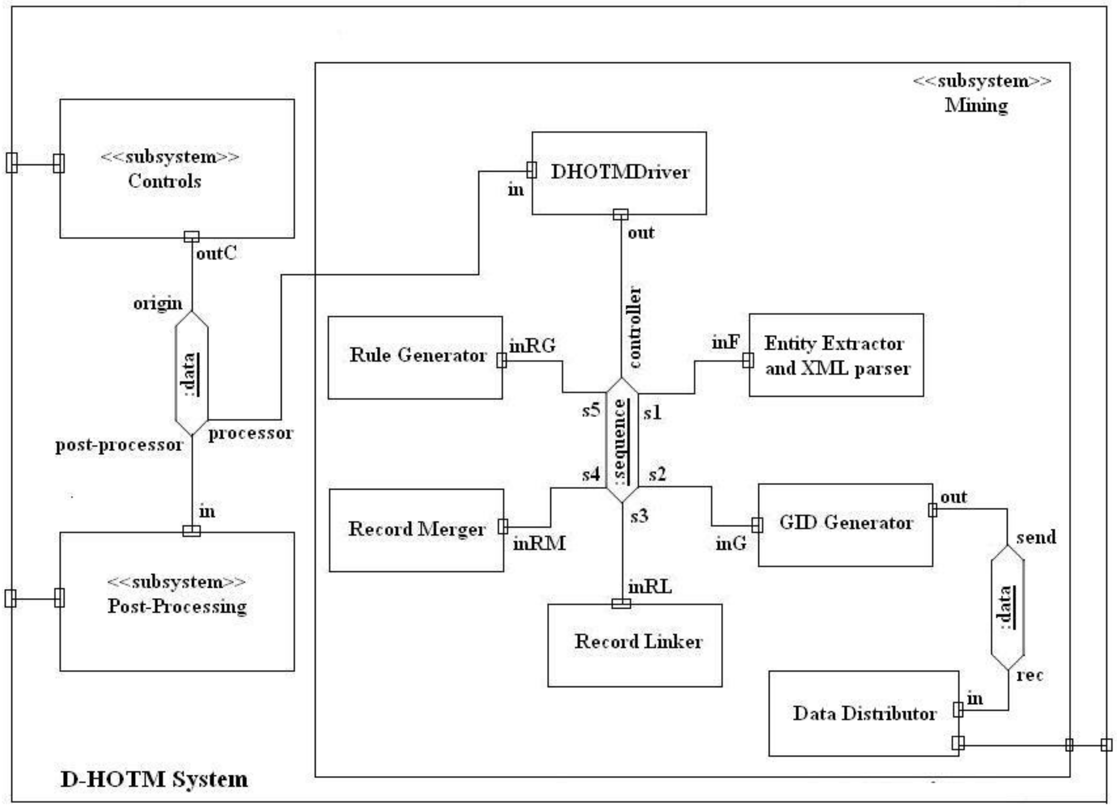


Fig. 1. D-HOTM Architecture

The Control and Post-Processing subsystems are implemented using Java and the Java Swing framework. The communication between these and the Mining subsystem is achieved through text files. The Control subsystem creates a parameter file (based on those parameters chosen by the user) that is read by a component in the Mining subsystem before it starts operating on the data. As operations complete, the Mining subsystem component writes the final and intermediate states of operation to XML files. These files are then used by the Post-Processing subsystem for further analysis by the user.

The Analysis component in this subsystem is responsible for providing the user with the ability to reason about the results generated by the system. This component receives as input the mined results produced by the D-HOTM system, discovers relationships in the results and provides the user with the ability to filter and browse through them. It also allows the user to save/transport results in various formats.

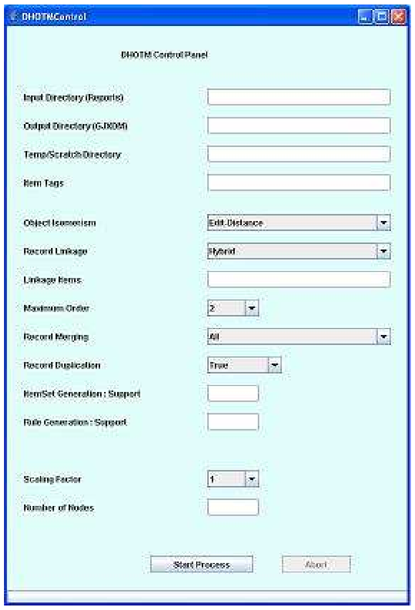
IV. Controls Component

The Controls component is the heart of the Controls subsystem in Fig. 1. This component provides the user with a GUI and writes the parameter file for the Mining subsystem. This component currently applies a very simple design, assisting in parameter file generation by allowing the user to type in text boxes or select from drop-down boxes. The user can specify three location parameters: where the input documents are located, where the entity extraction should output the XML files after extraction, and which directory should be used as a temporary directory when communicating with internal components of the system. In addition, the Controls component allows the user the ability to select or specify a number of other system parameters, such as different methods for record linkage, object isomerism techniques, approaches for merging of linked records, which linkage items should be used in record linkage, what should be the maximum order of association, and minimum support values used in itemset and rule generation. Plus, since this system is designed to be a research development platform, this component lets the user specify scaling factors for test input data and the number of nodes on which the system should execute in a distributed computing environment. Fig. 2 depicts a screenshot of the Controls GUI.

V. Entity Extraction and XML Parsing

The entity extraction phase is the first procedural step in the Mining subsystem in Fig. 1, and applies an extraction technique based on [20]. This technique, termed Reduced Regular

Expression (RRE) Discovery, discovers RREs for use in extraction of named entities. The algorithm discovers sequences of words and/or part-of-speech tags that, for a given entity, have high frequency in the labeled instances of the training data (i.e., true set) and low frequency in the unlabeled instances (i.e., false set). The algorithm first ascertains the most frequently appearing element of a reduced regular expression, called the root of the RRE. It then broadens the scope of the RRE in 'AND', 'GAP', and 'Start/End' learning phases (see Fig. 3 adapted from [20]).



Input Directory (Reports)	
Output Directory (GJXDM)	
Temp/Scratch Directory	
Item Tags	
Object Isomorism	Edit Distance
Record Linkage	Hybrid
Linkage Items	
Maximum Order	2
Record Merging	All
Record Duplication	True
ItemSet Generation : Support	
Rule Generation : Support	
Scaling Factor	1
Number of Nodes	

Fig. 2. Controls GUI

The approach uses a covering algorithm, implemented using RREs. Each RRE covers a portion of the true set for the particular feature being extracted. In the next step, the segments covered by the RRE are removed. This process, outlined in Fig. 3, repeats until all of the RREs are discovered that cover as many of the true segments as possible. A single RRE is discovered each time the process in Fig. 3 is completed, and is considered a sub-rule of the current entity. Finally, after all the processes have completed (i.e., all possible RREs have been discovered and all instances covered), the system uses an "OR" operator to combine the sub-rule RREs into a single rule that is also an RRE. This single rule can be used to extract the desired information from previously unseen data.

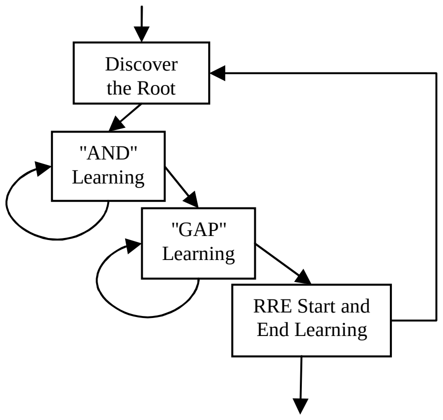


Fig. 3. Discovering Reduced Regular Expressions

The output from this Entity Extractor is Global Justice XML Data Model (GJXDM) formatted XML files. GJXDM aims to provide a consistent, extensible, maintainable XML schema reference specification for data elements and types, which represents the data requirements of the general justice and public safety communities. It also aims to provide a baseline model for the data dictionary, which can be represented in advanced technologies beyond XML Schemas. The GJXDM vision is to significantly advance justice information sharing by providing a common language and vocabulary that reduces cost and technical barriers [42].

This output is also compatible with the National Information Exchange Model (NIEM). The purpose of NIEM is to provide the foundation and building blocks for interoperable information sharing and data exchange at the national level. The NIEM project was formally announced at the GJXDM Executive Briefing on February 28, 2005. It is a joint venture between the U.S. Department of Homeland Security (DHS) and the U.S. Department of Justice (DOJ) with outreach to other departments and agencies. GJXDM is the base technology for NIEM. NIEM will leverage both the extensive GJXDM reference model and the comprehensive GJXDM XML-based framework and support infrastructure [43].

The D-HOTM system currently utilizes a subset of GJXDM and NIEM consisting of five basic entity types: Person, Location, Property, Organization, and Activity. Each basic entity type contains many sub-entities; for example, Person contains Person Name, Phone Number, Weight, Height, ID, and Date of Birth. Similarly, Property is divided into several sub-types such as Vehicle, Aircraft, Boat, etc., each sub-type with its own identification number and value information. In our system to date, we have mapped eight types of named entities into the GJXDM format, including weight, height, date of birth, drugs, street address, person name, social security number, and phone number.

Processing

The input to the Entity Extractor is a directory, which contains all the local textual data files (raw data) from which entities will be extracted. This directory is input by user in the Controls GUI, which is in the Controls subsystem, depicted in Fig. 1. The DHOTMDriver component in the Mining subsystem reads the input directory from the parameter file and sends it to the Entity Extraction component. The output from the Entity Extractor is a collection of GJXDM format files, one for each textual input file. The Entity Extractor first uses a part of speech tagger to tag the textual data, and then applies a series of RRE rules on the tagged data. These RRE rules are drawn from another training module, discussed in [20]. After applying the RRE rules, the named entities are extracted and written into a GJXDM format file. The D-HOTM system currently uses Perl to implement this Entity Extractor, but future development will employ a compiled language, to enhance execution speed and scalability.

The XML Parser component performs a data-format translation between the Entity Extractor and the following D-HOTM components, built upon the Text Mining Infrastructure (TMI) [44]. The TMI is an open-source framework designed for high-end, scalable text mining, and aims to provide a robust software core for research and development of text mining applications. Many data storage and retrieval classes available in the TMI are based on an Item-Feature framework (where an Item is defined as a collection of Features). The XML Parser parses the GJXDM data hierarchy, transforming sub-tags and values into a single attribute-value pair: the attribute being a concatenation of all sub-tags, and the value being the data at the leaf of the GJXDM hierarchy tree. These newly created GJXDM files are then sent to the next component in the system, the Global ID (GID) Generator.

VI. Global ID Generation

One of the strengths of the D-HOTM system is that it does not require establishment or knowledge of a global schema a priori. Still, many aspects of data mining require that some form of unique identifier be provided for various objects, especially when working with distributed sources. The role of the Global ID (GID) Generator component is to establish such identifiers.

One central aspect to ID assignment is the issue of object isomerism, determining whether or not two objects are the same. This issue is important in several steps of the Mining subsystem, including ensuring that distinct objects are given distinct identifiers (whereas the same objects should be assigned the same identifiers, even if at different locations), and also preventing “false positive” linkages between records. The record linkage process assumes that records can be linked through linkage items, or entities, by matching the items. Exact matching of linkage items (i.e., entities) is not possible in a variety of applications including those that involve, for instance, the use of names. For example, *I. Jones* and *Indiana Jones* may actually represent the same person. In this case, we need to identify unique linkage items to perform record linkage.

Although we have not developed our own method for mapping object identifiers, much work has been accomplished in a related research area traditionally referred to as record linkage. Record linkage is the process of comparing records from two or more data sources in an effort to determine which pairs of records represent the same object, and is most often applied to discover duplicate records [45]. This definition is somewhat different from the definition provided for the term “record linkage” in the D-HOTM theoretical framework (presented in Section VII below). In essence, the record linkage process in the D-HOTM framework links two records when they contain an item or items that serve to uniquely identify the same real world entity, while the traditional linkage process links two records when they represent the same real world entity. The difference, further elucidated in Section VII, boils down to the distinction between a record and an object. In the D-HOTM framework, objects are specified by the subset of items used as a global identifier (e.g., SSN or a combination of title and authors), while records contain items, or entities, extracted from documents (e.g., investigative police reports). The traditional definition of record linkage does not distinguish between objects and records in this sense – rather, records and objects are one and the same. As a result, the record linkage process in this framework is able to discover higher-order links between records using object identifiers, while the traditional linkage process aims only to discover duplicate records. To differentiate between these two definitions, in what follows record linkage refers the linkage process defined in Section VII, and we refer to the traditional record linkage process using the term object isomerism.

One widely studied approach to object isomerism is based on string comparison, and this is the approach taken in D-HOTM. An early string comparator is the Damerau-Levenstein (D-L)

Metric [46], [47], which is only one particular comparator metric from the class of edit distance metrics. The basic idea is that any string can be transformed into another string through a sequence of substitutions, deletions, insertions, etc. The smallest number of such operations required to change one string into another is a measure of the difference between them.

In general, the object isomerism process comprises three steps: candidate generation, similarity calculation, linkage and closure. The candidate generation step involves discovery of record pairs that are loosely similar (e.g., share common field values) so that evaluating all possible record pairs can be avoided. Similarity is then computed for every candidate pair of records identified in the candidate generation step. The final step – linkage and closure – links the candidate pairs with similarity higher than a threshold and forms the final equivalence classes using transitive closure.

Processing

The GID Generator begins by reading in the files from the XML Parser on a given local node in the distributed environment, storing the entities into a TMI Item Feature Set (IFS) for easy retrieval. Next, the IFS is traversed and entities are added to a GIDManager class, which stores regular and inverted indices relating objects to their assigned Global ID (GID). GIDs are currently created simply by taking the entity’s content and appending an additional ID tag, yet this may become more complex in future versions. With all local entities indexed, each local GIDGenerator in the distributed environment then prepares a GID list to be transmitted to all other distributed D-HOTM nodes, currently via MPI library calls. The GID lists are communicated to other nodes, each of which then merge these incoming lists with their own. Once all communication is complete, each distributed node then contains the same mapping of GIDs and their respective entities, thereby establishing a global ID schema for the current mining job. Also, the GIDGenerator shares records (i.e., collections of entities) in a similar manner to facilitate the record linking and merging by the next component.

VII. Record Linkage and Merging

As their respective names imply, the Record Linker component is responsible for determining which records possess higher-order links, and the Record Merger combines multiple linkable records into one. These are depicted in the Mining subsystem in Fig. 1. The link discovery process of the D-HOTM system provides its ability to recognize and harness the information in distributed higher-order links. Therefore, this section discusses in detail the theoretical framework underlying these link discovery algorithms.

Although much work has been done to explore higher-order associations in ARM, no effort has been devoted to a propositional higher-order ARM framework. This section will first define higher order links, and then present two algorithms to discover higher order itemsets based on different approaches to link analysis. A comparison of the two algorithms is also presented.

A. Higher Order Associations

ARM generates rules based on item co-occurrence statistics. Co-occurrence, also called 1st-order association, captures the fact that two or more items appear in the same context. Orders of association higher than 1st-order are termed higher-order associations. Higher-order association refers to association among items that come from different contexts. The higher-order associations are formed by linking different contexts through common item(s). For example, if one customer buys {milk, eggs}, and another buys {bread, eggs}, then {milk, bread} is a *2nd*-order association linked through "eggs". It is obvious that the 2nd-order association is reflexive, symmetric and transitive. Without loss of generality, we simplify the discussion in what follows by dealing only with transitive associations (i.e., we ignore reflexive and symmetric ones).

Definition 1: If item *a* and item *b* from different transactions can be associated across *n* distinct records, then items *a* and *b* are n^{th}-order associated, denoted as $a \sim^{r_1} i_1 \sim^{r_2} i_2 \sim \cdots \sim^{r_{n-1}} i_{n-1} \sim^{r_n} b$ where ~ represents the co-occurrence relation and *i* is termed a linkage item. The order of a higher-order association is determined by the number of distinct records *n*.

This framework allows each record to occur at most once in a given transitive link. Otherwise, cyclical links are possible such as $a \sim^{r_1} i_1 \sim^{r_1} i_2 \sim^{r_1} a$, which allows an item to be linked to itself at any order. This constraint is also necessary to be consistent with the original ARM framework. For example, given a higher-order association $a \sim^{r_1} i_1 \sim^{r_2} i_2 \sim^{r_1} b$, based on definition 1, *a* and *b* are 2nd-order associated because there are two distinct records in the link. This conflicts however with the fact that *a* and *b* actually are 1st-order associated since they both come from r_1.

The following theorem shows that higher-order links with repeated records can be transformed into a higher-order link per definition 1.

Theorem 1: For any higher-order link between two items, there must exist at least one link between those two items with no repeated records.

Proof: Suppose in higher order link $a \sim^{r_1} i_1 \sim^{r_2} i_2 \sim \cdots \sim^{r_{n-1}} i_{n-1} \sim^{r_n} b$ record $r_u = r_v$, then the higher order link could be written as $a \sim^{r_1} i_1 \sim^{r_2} ...i_{u-1} \sim^{r_u} i_u \sim^{r_{u+1}} \cdots \sim i_{v-1} \sim^{r_v} i_v ... \sim i_{n-1} \sim^{r_n} b$, therefore i_{u-1}, i_u, i_{v-1} and i_v come from the same record. Thus, $a \sim^{r_1} i_1 \sim^{r_2} ...i_{u-1} \sim^{r_u} i_v ... \sim i_{n-1} \sim^{r_n} b$ is a valid higher order link with no repeated records between item *a* and *b*.

B. Latent Higher Order Itemset Mining (LHOIM)

This subsection presents an algorithm to discover latent itemsets.

1) Definitions

Latent itemsets are itemsets in which item pairs may be associated by orders of one or higher. For example, the itemset *abk*, formed from the higher order link $a \sim^{r_1} b \sim^{r_2} c \sim \cdots \sim^{r_{n-1}} f \sim^{r_n} k$, is *latent higher order* associated: *ab* is 1st-order associated, *bk* is *n-1th*-order associated, and *ak* is *nth*-order associated. Due to these associations, *abk* is a latent itemset.

A *link group* is a group of higher order links written as $r_1 \sim^{I_1} r_2 \sim^{I_2} \cdots \sim^{I_{n-1}} r_n, I_j = r_j \cap r_{j+1}$. To simplify this notation, henceforth $r_1 \sim r_2 \sim \cdots r_{n-1} \sim r_n$ is used to represent a link group. Clearly, the latent itemsets generated from the higher order link $a \sim^{r_1} b \sim^{r_2} c \sim \cdots \sim^{r_{n-1}} f \sim^{r_n} k$ could also be discovered from the link group $r_1 \sim r_2 \sim \cdots r_{n-1} \sim r_n$. The *order* of a link group is the number of records it contains. The *size* of a link group can be calculated by taking the product of the sizes of I_j. For example, given a 3rd-order link group L$_g$: $r_1 \sim^{acd} r_2 \sim^{ef} r_3$, the number of links in L$_g$ is *size(*L$_g$*)*=|*acd*|*|*ef*|=6. The *size* of the link group is therefore defined as $\prod_{i=1..n-1}(|r_i \cap r_{i+1}|)$.

Link groups aid in the discovery of higher order itemsets because for a given higher order link $a \sim^{r_1} b \sim^{r_2} c \sim \cdots \sim^{r_{n-1}} f \sim^{r_n} k$, there may exist many higher order links that share the same record sequence. The number of such links is $\prod_{i=1..n-1}(|r_i \cap r_{i+1}|)$. All these links will result in the same set of latent itemsets. Thus, instead of dealing with $\prod_{i=1..n-1}(|r_i \cap r_{i+1}|)$ higher order links, a single link group accomplishes the same goal.

Having defined the context of latent itemsets in link groups, the next issue is generating those contexts. If each record is mapped to a node, and edges are mapped to common shared items, then the problem of finding all link groups reduces to finding all simple paths between two vertices in a graph. Finding simple paths between two vertices is solved using a backtracking technique whose pseudocode is shown in Fig. 5.

Suppose latent itemset *A* is generated from link group $r_1 \sim r_2 \sim \cdots r_{n-1} \sim r_n$. Following the Apriori method would provide the support of *A* as the number of its appearances in the link group. The link group contains $\prod_{i=1..n-1} (|r_i \cap r_{i+1}|)$ links, each of which can generate the latent itemset *A*, and thus the support of *A* is $\prod_{i=1..n-1} (|r_i \cap r_{i+1}|)$. This method presents two challenges: it results in very large numbers for the support of the latent itemsets, and it ignores the effect of the order. To address these issues, in what follows a metric is presented to calculate support for latent itemsets which leverages the size of the link group as well as the effect of the order.

Let *L(A)* be the set of link groups which contain the latent itemset *A*. We define the *support* of a latent itemset *A* as:

$$\sum_{m=1}^{\max\_order} \log_{10}\Big(\sum_{l \in L(A), l.order=m} l.size + 1\Big) / m \qquad \text{(Equation 1)}$$

The sum in Equation 1 results from the fact that since the same latent itemset can be generated at different orders, the *global* support for a given latent itemset must include the *local* support at each order *m*. Thus $\sum_{l \in L(A), l.order=m} l.size$ form the *local* component of support for an itemset at a specific order *m*. The idea behind global support is simply to account for both the number of higher-order links supporting a given latent itemset as well as the order of the itemset. As order grows, intuition suggests that support ought to decrease – thus the denominator *m*. This reflects the assumption that the longer the link between records, the weaker the itemset association. In contrast, intuition also suggests that the more link groups that contain a given itemset, the stronger the support should be. These two intuitions are just that – certainly, extensive experimentation is required to ascertain the utility of this definition of support. The challenge arises when one considers the exponential growth of the link groups' sizes given that order grows linearly. Again, intuition suggests that both of these factors are equally important. Thus, in order to constrain *l.size* to grow linearly with order, the $\log_{10}$ is taken. Also, one is added to *l.size* in the numerator to ensure that the argument to $\log_{10}$ is non-zero.

2) Algorithm

Based on the framework discussed in previous subsection, an algorithm to discover latent-order itemsets in presented herein.

TABLE I.

NOTATION FOR LATENT ITEMSET MINING ALGORITHM

MR	Set of merged records. Each member of this set has three fields: i) record, ii) size and iii) order.
L_k	Set of large *k*-itemsets. Each member of this set has two fields: i) itemset and ii)support count.
C_k	Set of candidate *k*-itemsets. Each member of this set has two fields: i) itemset and ii)support count.

Latent Itemset Mining

Input: *D, L, max_order, minsup*

Output: latent itemsets

```
1.  form adjacency list
2.  for each G
3.      Enumpath(G, max_order)
4.  for each n^th-order linkgroup lg: $r_1 \sim r_2 \sim \cdots r_{n-1} \sim r_n$
5.        $r = \bigcup_{i=1}^{n} r_i$ , $r.size = \prod_{i=i}^{n-1} (|r_i \cap r_{i+1}|)$ , $r.order = n$
6.        add r to MR
7.  $L_1$ = {large 1-itemsets};
8.  for ( k=2; $L_{k-1}$!=null; k++)
9.        $C_k$ = apriori-gen($L_{k-1}$);
10.       for each R do
11.           $C_t$ = subset($C_k$, R)
12.           for all candidates $c \in C_t$
13.                   c[R.order].count+=R.size
14.       for all candidates $c \in C_x$
15.           c.sup = $\sum_{order} \frac{\log 10(c[order].count + 1)}{order}$
```

```
16.     $L_k = \{ c \in C_k \mid c.sup \geq minsup \}$
17.  Answer = $\bigcup_k L_k$ ;
```

Fig. 4. Latent Itemset Mining Algorithm

The first step is to form a connected undirected graph from the records, where each vertex represents a record, an edge between two vertices exists when two records share at least one common item, and the weight of the edge represents the number of common items. The generated graph may contain more than one connected subgraph.

Step 3 – Enumpath – discovers all the higher order link groups for orders up to *max_order* for the connected subgraph G. Enumpath, described in Fig. 5, employs the algorithm in [48] to find all simple paths (i.e., link groups) between two vertices. The worst case time complexity of this step is $O(|V||E|)$ for a given path where V is the set of vertices and E the edges in G [48].

```
Enumpath
Input: G, L, max_order
Output: higher-order link groups

for t=$r_1$ to $r_{n-1}$
   for j=0 to AdjacencyList[i].size
      v=AdjacenyList[i].at[j]
         enupath(t, v, max_order)
```

Fig. 5. Enumpath Algorithm

Steps 4 to step 6 in Fig. 4 generate the merged record from each link group. The size and order of each link group are passed with the merged record. Steps 7 to 17 of Fig. 4 discover the frequent latent itemsets. The level-wise process is similar to Apriori, except for the calculation of support. The support calculation for a given latent itemset comprises two steps: first, the sizes of the merged record supporting the latent itemset are kept per order, as shown in step 13; then, the final support count is calculated based on the aforementioned metric, as shown in step 15. Those latent itemsets which meet the threshold become the frequent latent *k*-itemsets and are used to generate the latent *k+1*-itemsets.

C. Higher Order Itemset Mining (HOIM)

This subsection discusses Higher Order Itemset Mining (HOIM) which is based on a different definition for higher order itemsets.

1) Definitions

In HOIM, a *higher-order itemset* is defined in a different manner. In effect, the higher order associations in itemsets are not latent, but explicit.

Definition 2: An n^{th}-order k-itemset is a k-itemset for which each pair of items is n^{th}-order associated. For example, if *abc* is a 3rd-order itemset, then there must exist at least three 3rd-order associations between a and b, b and c and a and c respectively. Definition 3 introduces the concept of a *higher-order recordset*, which is the context of higher-order itemsets.

Definition 3: An n^{th}-order k-recordset is a k-recordset where there exists at least one n^{th}-order link group between each record pair.

A n^{th}-order itemset $i_1i_2\dots i_n$ is supported by a n^{th}-order recordset $r_1r_2\dots r_n$ if no two items originate from the same record. For example, given the following three records $<r_1$, *abc*$>$, $<r_2$, *adef*$>$, $<r_3$, *bfgh*$>$, then $r_1r_2r_3$ is a *2nd*-order recordset due to $r_1 \sim r_2$, $r_1 \sim r_3$ and $r_2 \sim r_3$. Since $b \in r_1$, $e \in r_2$ and $g \in r_3$, *beg* is a *2nd-order* itemset supported by the *2nd-order* recordset $r_1r_2r_3$.

Similar to link groups, the *size* of a recordset is calculated by taking the product of the sizes of each link group. It is important to note that a given recordset might be composed of different link group combinations. This may happen for nth-order recordsets when n is greater than two. For example, Fig. 6.a and Fig. 6.b represent the same 3rd-order 3-recordset $r_1r_2r_3$ formed in two ways using different link groups. The total size of $r_1r_2r_3$ must incorporate both recordsets. Generalizing from this example, given j instances of n^{th}-order k-recordset rs, its size is defined as:

$$size_{n\_k}(rs) = \sum_{u=1}^{j} (\prod_{v=1}^{k(k-1)/2} \lg_v .size) .$$

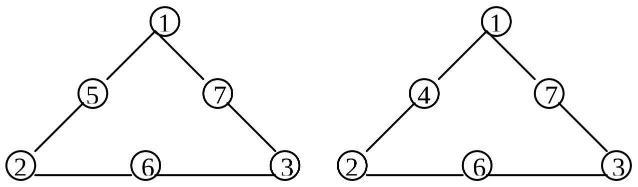


Fig. 6. Two forms of Recordset $r_1r_2r_3$

Given the sizes for all recordsets, the support of a *k*-itemset *is* is defined as:

$$\sup_k(is) = \sum_{t=1}^{\max\_order} \frac{\log_{10} \sqrt{\sum size_{n\_k}(rs) + 1}}{t} \quad \text{(Equation 2)}$$

This metric is similar to Equation 1 – the global support is calculated by adding the local support at each level. The local support is also designed based on the intuition that the size of the recordsets should be of same importance as the order. To constrain *size*$_{n-k}$*(rs)* to grow linearly with order, first the square root of *size*$_{n-k}$*(rs)* is taken and then the $\log_{10}$. The square root accounts for the O(n^2) growth of number of edges in a recordset as order grows; the $\log_{10}$ accounts for the exponential growth of *size*$_{n-k}$*(rs)*. As before, one is added to *size*$_{n-k}$*(rs)* in the numerator to ensure that the argument to $\log_{10}$ is non-zero.

2) Algorithm

This section presents the Higher Order Itemset Mining algorithm, which discovers higher order itemsets. HOIM is structured in an order-first level-wise manner. Level-wise means that the size of *k*-itemsets increases in each iteration (as is the case for Apriori), while order-first means that at each level, itemsets are generated across all orders. Notation used in the algorithm is provided in Table II, followed by the algorithm itself.

TABLE II.

NOTATION FOR HIGHER ORDER ITEMSET MINING ALGORITHM

RS_k	Set of *k*-recordsets. Each member has: i) recordset; ii) order; iii) size of the recordset.

RS_{n_k}	Set of n^{th}-order k-recordsets. Each member has: i) recordsets; ii) size.
IS_k	Set of k-itemsets. Each member has: i) itemset; ii) global support of the itemset.
IS_{n_k}	Set of n^{th}-order k-itemsets. Each member has: i) itemsets; ii) local support.

```
Higher Order Itemset Mining
Input: D, L, max_order, minsup
Output: higher order itemsets
1.  form Adjacency List
2.  for each connected subgraph G
3.    Enumpath(G, max_order)
4.  for each $n^{th}$-order linkgroup $l$: $r_1 \sim r_2 \sim \cdots r_{n-1} \sim r_n$
5.       $rp=(r_1, r_n)$,  $rp.size = \prod_{i=i}^{n-1}(|r_i \cap r_{i+1}|)$ , $rp.order = n$,
6.       if $RS_2(rp, rp.order)$ is valid
7.            $RS_2(rp, rp.order).size$ += $rp.size$
8.       else $RS_2(rp,order)=size$
9.  for ( $k$ = 3; $RS_{k-1}\neq\phi$; $k$++)      // k: size of the recordset
10.   for ( $n$ = 2; $n$ < max_order; $n$++)  // n: order
11.      $RS_{n\_k}$ = Gen_RS($RS_{n\_k-1}$ );
12.      for each recordset $rs \in RS_{n\_k}$
13.         Enum_IS($rs$, $n$);
14.   for each itemset $is$ where $|is|=k$
15.         $IS_k(is).\text{sup}=\sum_{t=2}^{\max\_order} \log_{10}\sqrt{IS_{n\_k}(is).\text{sup}+1}/u$
16.   Answer = Answer $\cup$ { $is$ | $IS_k(is)$.sup >= $min\_sup$ }
```
Fig. 7. Higher Order Itemset Mining Algorithm

The first three steps of *HOIM* are the same as *LHOIM* – to generate all possible link groups. For each link group, step 5 generates the corresponding *2*-recordsets with the size and order information. Steps 6 to 8 calculate the size of a *2*-recordsets at a given order and store it in RS_2.

Steps 9 through 16 of the *HOIM* algorithm in Fig. 7 comprise one outer and two inner loops. The outer loop proceeds in a level-wise manner and keeps track of the sizes of recordsets. Although the (*k+1*)-recordsets are generated in an Apriori-like fashion based on *k*-recordsets from the previous iteration, no pruning is performed for recordsets. The first inner loop, from steps 3 through 6, proceeds order-first and generates different order itemsets from the *k*-

recordsets, and calculates their corresponding support values. Fig. 9 depicts the order-first level-wise structure of our higher order itemset mining algorithm.

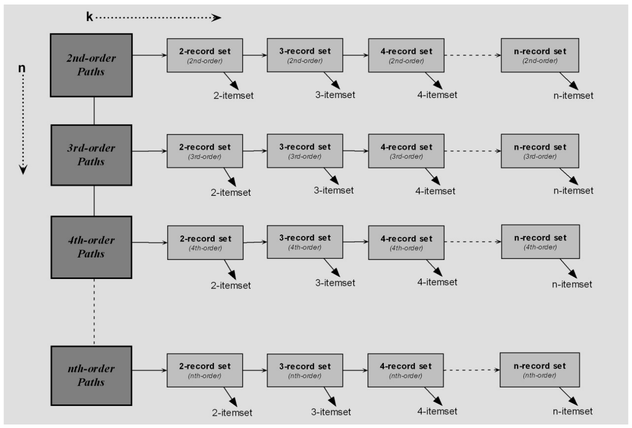


Fig. 9. Order-First Level-wise Structure

In more detail, step 11 in Fig. 7 generates the n^{th}*-order k-recordsets* based on the n^{th}*-order (k-1)-recordsets* using Apriori's candidate generation ability. The process is shown in Fig. 8. The size of the recordset is calculated based on the equation for $size_{n\text{-}k}(rs)$ above in Section VII. C.

```
Gen_RS ($RS_{n\_k}$)
1.    $RS_{n\_k+1} = apriori\_gen(RS_{n\_k})$
2.    for each recordset $rs \in RS_{n\_k+1}$
3.        for each $rp$ in $rs$
4.            $rs.size \times= RS_2(rp,n).size$
```

Fig. 8. Generate Recordset

For each n^{th}*-order k-recordset* generated, step 13 of Fig. 7 enumerates all possible n^{th}*-order k-itemsets* from the recordset. Fig. 10 provides the pseudocode for this step.

```
Enum_ IS ($rs$, $n$)
1.  $k$ = number of records in the $rs$
2.  pick one item from each record
3.      if the items are different
4.          $IS_{n\_k}(is).sup += rs.size$
```

Fig. 10. Enumerating Itemsets from a Recordset

Steps 14 and 15 of Fig. 7 calculate the global support for a single *k*-itemset across orders from two to *max_order* based on the support in Equation 2. If the global support meets the threshold, the *k*-itemset is added to the final output in step 16.

D. Comparison of Two Approaches, LHOIM and HOIM

LHOIM generates latent higher order itemsets while HOIM generates explicit higher order itemsets. This subsection theoretically and empirically compares these two approaches.

Latent itemsets are formed from link groups that do not contain repeated records. On the other hand, explicit itemsets are formed from recordsets, which are composed of same-order link groups. Before comparing the two kinds of itemsets, one issue needs to be addressed: for a recordset composed of link groups, should we allow the link groups to share common records? If the sharing of common records amongst link groups is disallowed, then one can claim that for the same input, the latent itemsets generated from LHOIM will be a superset of the explicit itemsets generated from HOIM. This is proven in Theorem 2, following Lemma 1 below.

Lemma 1: For a m-clique, there are $(m\text{-}1)!/2$ Hamiltonian cycles and $m!/2$ Hamiltonian paths.

Proof: (1) $m=3$: a triangle with one Hamiltonian cycle. (2) assume that for any m, lemma 1 is true; (3) for the $m+1$ case: take the graph including one additional vertex added to the previous m vertices. For m vertices, all the Hamiltonian cycles have already been found. Any single m-Hamiltonian cycle, say ($v1$, $v2$, …., v_m, $v1$), contains m edges. Randomly select any edge, say (v_i, v_{i+1}), and break it to be (v_i, v_{m+1}, v_{i+1}). The cycle now becomes a (m+1)-Hamiltonian cycle. This can be done for each edge of the m-cycle. Thus, for each m-Hamiltonian cycle, there are $m(m+1)$ Hamiltonian cycles. Thus, $f(m+1)=m*f(m)=m*(m\text{-}1)!/2=((m+1)\text{-}1)!/2$. For any Hamiltonian cycle with n nodes, n Hamiltonian paths can be generated. Thus, for m-cliques with $(m\text{-}1)!/2$ Hamiltonian cycles, $m!/2$ Hamiltonian paths can be generated.

Theorem 2: Using the same input data, the itemsets generated from the LHOIM will be a superset of the itemsets generated from HOIM, when no common records are allowed in the link groups composing the recordset.

Proof: Since the link groups in a recordset do not share any common records, the recordset thus forms a clique. Per Lemma 1, for any *k*-recordset, there exist *k*!/*2* link groups, covering all *k* records in the recordset. Thus, any explicit *k*-itemset generated from the *k*-recordset will be generated from the link group as well. On the other hand, the latent itemsets generated from link groups might not exist in the explicit itemsets from the *k*-recordset, since the explicit itemsets are limited to one item from each record only.

However, if link groups in a given recordset are allowed to share common records, then the claim of Theorem 2 does not hold. For example, suppose the 3rd-order 3-recordset $r_1r_2r_3$ is formed based on the following link groups: $r_1 \sim r_0 \sim r_2$, $r_1 \sim r_0 \sim r_3$ and $r_2 \sim r_0 \sim r_3$, all containing r_0. The recordset can be represented as in Fig. 11, and clearly there is no Hamiltonian path in this graph. In this situation, it not possible to form a link group covering r_1, r_2 and r_3 together.

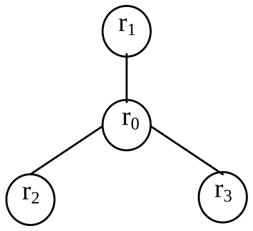


Fig. 11. A Sample Recordset

In order to explore different types of higher order itemsets, the current implementation of the HOIM algorithm allows the link groups in a given recordset to share common records.

As different support metrics in two algorithms are applied, it is interesting to consider whether any theoretical conclusions can be drawn in terms of support. For example, given a recordset and those link groups covering the same records, when the recordset has four or less records, the support in HOIM is always less than the support in LHOIM. However, as the number of records increases, the support in HOIM scales faster than that in LHOIM. Continued exploration and evaluation of these and other support metrics forms part of the future work.

E. Sample Run

To further illustrate this algorithm, this section provides a simple example simulating a real problem: buying computer parts and a book about building computers from an e-Marketplace. There are four records in the database and five products. The five products are: Microsoft Wheel Optical Mouse (b); Building a PC for Dummies, Fifth Edition (c); Microsoft Internet Keyboard (d); Build the Ultimate Custom PC (e) and AMD Athlon 64 X2 (f). In Tables III-X these items are referred to using their letter designation (b, c, d, e and f). Table III summarizes the input.

First the transaction database is converted to a graph representation and passed to the two algorithms. The simple paths (or link groups) discovered by *Enumpath* are shown in Table IV with the corresponding size values.

In the HOIM algorithm, the recordsets and corresponding itemsets of different sizes are generated per Fig. 7. As portrayed below, 2-recordsets are directly obtained from the *Enumpath* output, and 2-itemsets are generated from these 2-recordsets. Recordsets are generated in a level-wise manner, similar to Apriori as shown in Table V. n^{th}-order k-itemsets are generated from corresponding n^{th}-order k-recordsets. For each itemset generated, the local support is calculated using the formula derived in Section VII. C, at each order. After generating all orders of k-itemsets and their local support (see Tables VI-VIII), the global support is calculated for each itemset per Equation 2 in Section VII. C.

Instead of generating recordsets in a level-wise manner, LHOIM merges the records in the link groups directly. For each merged record, the order and size of the corresponding link groups are retained to calculate the support for the itemset as shown in the LHOIM algorithm in Fig. 4. Table IX shows the support for each itemset as well as the support at each order.

Table X compares the itemsets and the support values for Apriori, HOIM and LHOIM.

After all itemsets are collected, association rules are discovered using a standard ARM algorithm such as Apriori. This example demonstrates that both LHOIM and HOIM discover novel information. For example, both approaches discover the rule $ce \rightarrow f$, which means if a person buys the two books “Building a PC for Dummies, Fifth Edition” and “Build the Ultimate Custom PC,” they may very well be interested in “AMD Athlon 64 X2.” As the original database does not have any record containing the higher-order itemset *cef*, traditional ARM algorithms cannot discover this rule.

In the following section we describe the Analysis component of the D-HOTM system, and then proceed to discuss our experimental results.

TABLE III.

SAMPLE DATA

RecID	Itemsets
1	b,c
2	c,d,e
3	c,f
4	d,e,f

TABLE IV.

LINK GROUPS GENERATED FROM THE SAMPLE DATA

2nd-order paths	Size	3rd-order	Size	4th-order	Size
1-2	1	1-3-2	1	1-3-4-2	2
1-3	1	1-2-3	1	1-2-4-3	2
2-3	1	1-2-4	2	1-2-3-4	2
2-4	2	1-3-4	1	1-3-2-4	2

3-4	1	2-1-3	1	2-1-3-4	2
		2-4-3	2	3-1-2-4	2
		2-3-4	2		
		3-2-4	2		

TABLE V.

THE NTH-ORDER K-RECORDSETS GENERATED BY HOIM

k=2, n=2			k=3, n = 2					
1,2	bcde	1	1,2,3	bcdef	1			
1,3	bcf	1	2,3,4	cdef	2			
2,3	cdef	1						
2,4	cdef	2						
3,4	cdef	1						
k=2, n = 3			k=3, n = 3			k=4, n = 3		
1,2	bcde	1	1,2,3	bcdef	3	1,2,3,4	bcdef	18
1,3	bcf	1	1,2,4	bcdef	3			
1,4	bcdef	3	2,3,4	cdef	6			
2,3	cdef	3	1,3,4	bcdef	6			
2,4	cdef	1						
3,4	cdef	2						
k=2, n = 4			k=3, n = 4					
1,2	bcde	2	1,2,4	bcdef	6			
1,3	bcf	2	1,3,4	bcdef	12			
1,4	bcdef	3						
2,4	cdef	1						
3,4	cdef	2						

TABLE VI.

2-ITEMSETS IN HOIM

k=2				
	order = 2	3	4	Total
bc	0.12	0.08	0.09	0.29
bd	0.08	0.12	0.10	0.29
be	0.08	0.12	0.10	0.29
bf	0.08	0.12	0.10	0.29
cd	0.19	0.17	0.12	0.49
ce	0.19	0.17	0.12	0.49
cf	0.19	0.17	0.12	0.49
de	0.12	0.05	0.04	0.21
df	0.17	0.14	0.08	0.39
ef	0.17	0.14	0.08	0.39

TABLE VII.

3-ITEMSETS IN HOIM

k=3				
	order = 2	3	4	Total
bcd	0.08	0.19	0.16	0.42

bce	0.08	0.19	0.16	0.42
bcf	0.08	0.19	0.16	0.42
bde	0.00	0.10	0.11	0.21
bdf	0.08	0.19	0.16	0.42
bef	0.08	0.19	0.16	0.42
cde	0.12	0.17	0.11	0.39
cdf	0.15	0.21	0.16	0.52
cef	0.15	0.21	0.16	0.52
def	0.12	0.14	0.00	0.26

TABLE VIII.

4-ITEMSETS IN HOIM

k=4				
	order = 2	3	4	Total
bcde		0.21		0.21
bcdf		0.21		0.21
bcef		0.21		0.21
bdef		0.21		0.21
cdef		0.21		0.21

TABLE IX.

ITEMSETS IN LHOIM

	2nd-order	3rd-order	4th-order	Total
b	0.24	0.28	0.28	0.80
c	0.42	0.37	0.28	1.07
d	0.39	0.37	0.28	1.04
e	0.39	0.37	0.28	1.04
f	0.39	0.37	0.28	1.04
bc	0.24	0.28	0.28	0.80
bd	0.15	0.28	0.28	0.71
be	0.15	0.28	0.28	0.71
bf	0.15	0.28	0.28	0.71
cd	0.39	0.37	0.28	1.04
ce	0.39	0.37	0.28	1.04
cf	0.39	0.37	0.28	1.04
de	0.39	0.37	0.28	1.04
df	0.35	0.37	0.28	1.00
ef	0.35	0.37	0.28	1.00
bcd	0.15	0.28	0.28	0.71
bce	0.15	0.28	0.28	0.71
bcf	0.15	0.28	0.28	0.71
bde	0.15	0.28	0.28	0.71
bdf	0.00	0.28	0.28	0.56
bef	0.00	0.28	0.28	0.56
cde	0.39	0.37	0.28	1.04
cdf	0.35	0.37	0.28	1.00
cef	0.35	0.37	0.28	1.00
def	0.35	0.37	0.28	1.00
bcde	0.15	0.28	0.28	0.71

bcdf	0.00	0.28	0.28	0.56
bcef	0.00	0.28	0.28	0.56
bdef	0.00	0.28	0.28	0.56
cdef	0.35	0.37	0.28	1.00
bcdef	0.00	0.28	0.28	0.56

TABLE X.

COMPARISON

	Apriori	HOIM	LHOIM
b	1		0.8
c	3		1.07
d	2		1.04
e	2		1.04
f	2		1.04
bc	1	0.29	0.8
bd		0.29	0.71
be		0.29	0.71
bf		0.29	0.71
cd	1	0.49	1.04
ce	1	0.49	1.04
cf	1	0.49	1.04
de	2	0.21	1.04
df	1	0.29	1
ef	1	0.29	1
bcd		0.42	0.71
bce		0.42	0.71
bcf		0.42	0.71
bde		0.21	0.71
bdf		0.42	0.56
bef		0.42	0.56
cde	1	0.29	1.04
cdf		0.49	1
cef		0.49	1
def	1	0.26	1
bcde		0.21	0.71
bcdf		0.21	0.56
bcef		0.21	0.56
bdef		0.21	0.56
cdef		0.21	1
bcdef			0.56

VIII. ANALYSIS

The Analysis component serves to assist the user in managing, exploring and experimenting with the wealth of information (comprised of itemsets, recordsets, association rules and other data) generated by the D-HOTM system. A component belonging to the Post-Processing subsystem, the Analysis component is a collection of a several graphical user interface (GUI) classes and a number of Java classes to support the GUI. These support classes include a set of Reader classes. The specialized Reader classes read in the state of the D-HOTM objects from the XML-style files output by the Mining subsystem. Each object type in the C++ mining code has a corresponding Java object in this component, with additional attributes and functionality. The Readers are used by a set of corresponding Manager classes, which are responsible for the storage and management of D-HOTM objects.

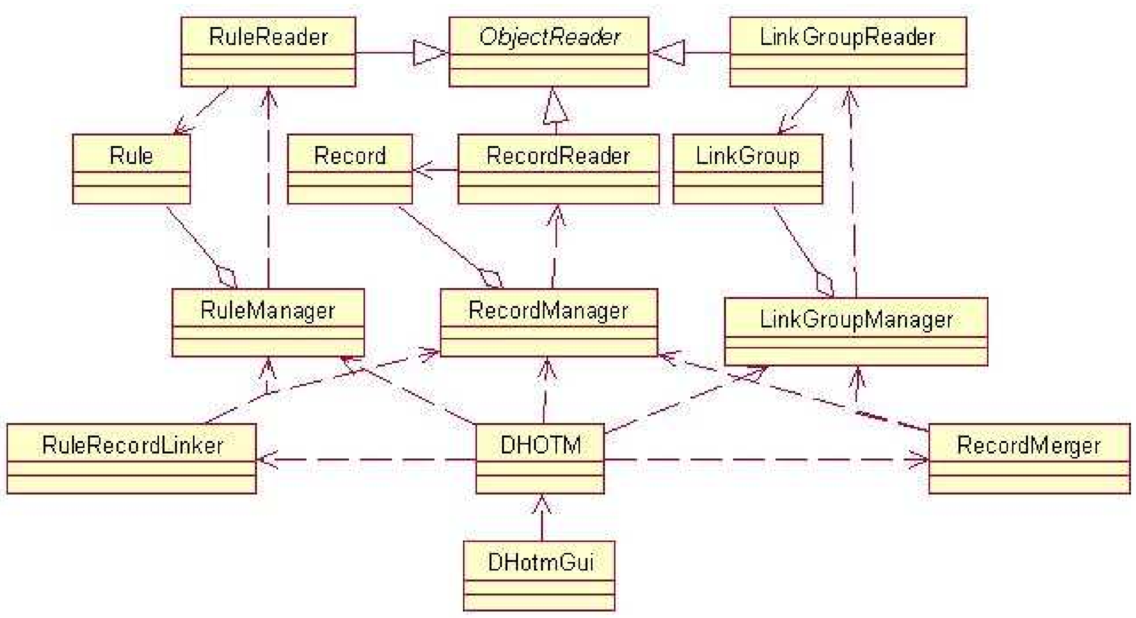


Fig. 12. UML Class Diagram of the Analysis Component

The Mining subsystem discovers linkable records and merges them based on link groups. The merged records are then used to derive rules that span multiple records. However, the processing components do not maintain the details of the rule generation process. To be able to provide the user the ability to analyze every aspect of the D-HOTM process, the Analysis component needs functionality to help put the different pieces together.

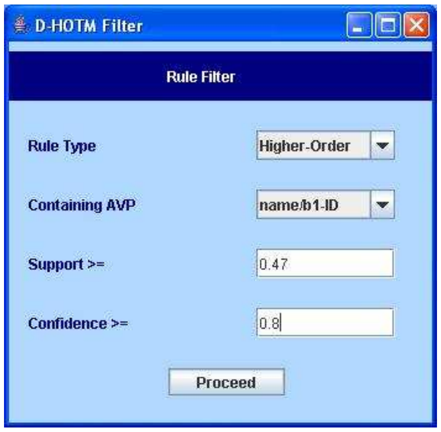
The Analysis component generates a graph structure from the information provided from the

D-HOTM XML output files. In the Mining subsystem, Rules are the final output, while Link Groups and Merged Records are intermediate results. In the graph structure, the Analysis component incorporates the concept of Successors and Predecessors. Although rules have no successors in this graph, they do have Records (original or merged) as their predecessors, and a Merged Record has a LinkGroup as its predecessor. These relationships are recovered by the Java classes LinkGroup, RecordMerger and RuleRecordLinker. The intention is to generate links of Rule ↔ Record ↔ LinkGroup, to be able to reach back to the origin of a particular rule. In such manner, the user can understand how a particular rule was generated and how its support value was assigned.

The GUI of this component thus provides the user the ability to traverse the directed graph of rules, records, merged records and link groups. The interaction with the user begins with the system asking the user to provide the type of rules required and other filter parameters of interest. With this information, this component reads the required set of rules and creates the graph.

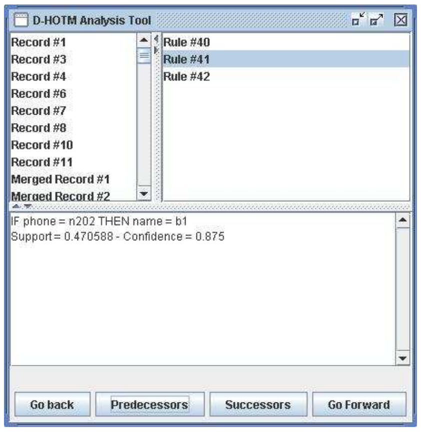
The GUI also provides the user with the functionality of moving to a predecessor or successor of a graph element, similar to the Columns view of the Finder application in Mac OS X. Graph elements of a given type are presented to the user in the form of a list. The user is also provided a view of two additional lists, the predecessor list and successor list. When the user selects a particular element in a list, a description of the element shows up. The user can then choose to either move to a predecessor or successor of this selected element. With any such move, the predecessor and successor lists are appropriately repopulated.

Moving arbitrarily in a directed graph can be confusing to a user. The user might take an undesired route and want to retrace a path. Also, in some cases, the user might want to explore several different routes in the graph. The GUI facilitates these common activities by providing "Back" and "Front" buttons, similar to the ones prevalent in modern-day browsers.



Rule Type: Higher-Order
Containing AVP: name/b1-ID
Support >= 0.47
Confidence >= 0.8

(a) Rule Filer

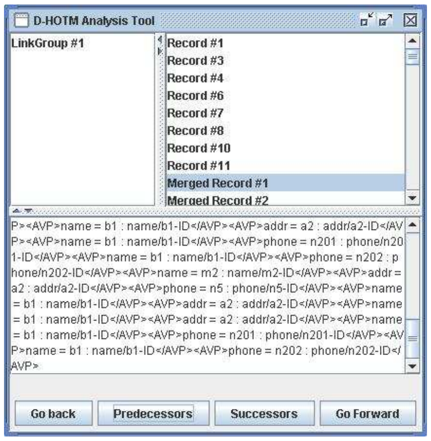


```
Record #1
Record #3
Record #4
Record #6
Record #7
Record #8
Record #10
Record #11
Merged Record #1
Merged Record #2
```

```
Rule #40
Rule #41
Rule #42
```

```
IF phone = n202 THEN name = b1
Support = 0.470588 - Confidence = 0.875
```

(b) Displaying Filtered Rules

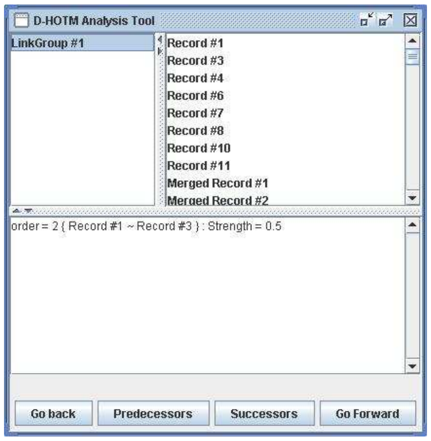


```
LinkGroup #1
```

```
Record #1
Record #3
Record #4
Record #6
Record #7
Record #8
Record #10
Record #11
Merged Record #1
Merged Record #2
```

```
P><AVP>name = b1 : name/b1-ID</AVP><AVP>addr = a2 : addr/a2-ID</AV
P><AVP>name = b1 : name/b1-ID</AVP><AVP>phone = n201 : phone/n20
1-ID</AVP><AVP>name = b1 : name/b1-ID</AVP><AVP>phone = n202 : p
hone/n202-ID</AVP><AVP>name = m2 : name/m2-ID</AVP><AVP>addr =
a2 : addr/a2-ID</AVP><AVP>phone = n5 : phone/n5-ID</AVP><AVP>name
 = b1 : name/b1-ID</AVP><AVP>addr = a2 : addr/a2-ID</AVP><AVP>name
 = b1 : name/b1-ID</AVP><AVP>addr = a2 : addr/a2-ID</AVP><AVP>name
 = b1 : name/b1-ID</AVP><AVP>phone = n201 : phone/n201-ID</AVP><AV
P>name = b1 : name/b1-ID</AVP><AVP>phone = n202 : phone/n202-ID</
AVP>
```

(c) Displaying Contents of a Record



```
LinkGroup #1
```

```
Record #1
Record #3
Record #4
Record #6
Record #7
Record #8
Record #10
Record #11
Merged Record #1
Merged Record #2
```

```
order = 2 { Record #1 ~ Record #3 } : Strength = 0.5
```

(d) Displaying Contents of a Link Group

Fig. 13. Filtering Rules using the Analysis GUI

Fig. 13.a shows the filtering process achieved by the Analysis component. Here, the user selects Higher-Order rules for analysis, and can also specify the entity (specified as an attribute-value pair, or AVP) of interest, support and confidence levels. Based on these parameters, the Analysis component displays only those rules satisfying these constraints, shown in Fig. 13.b. Specifically, these rules are higher-order association rules containing {number=b1} and have support and confidence greater than or equal to 0.47 and 0.8 respectively. When the user clicks on Rule#41 the GUI shows the content of the rule in the text-area located at the bottom of Fig. 13.b. Clicking on Rule#41 and the “Predecessors” button, the left column of the GUI is now populated with a number of record identifiers. All these records contain Rule#41 and hence serve as predecessors of this rule. Clicking on any of these records displays their contents in the text-area located at the bottom of the GUI, as shown in Fig. 13.c. The GUI provides information on both kinds of records: original records, identified with text starting with "Record#", and merged records, identified with text prefaced by "MergedRecord#".

Clicking on MergedRecord#1 and the “Predecessors” button displays the link group used to generate this merged record. In the case of original records, no predecessors would appear on the left column in the GUI.

Information regarding link groups is also available, as shown in Fig. 13.d, the result of clicking on LinkGroup#1. The text appearing in the lower text-area states which records are linked together to form this link group along with the strength of the link group.

IX. Experimental Results

Experiments were conducted to further evaluate the D-HOTM system on the National Center for Supercomputing Applications (NCSA) Tungsten Supercluster (Xeon Linux). Tungsten is composed of Intel 3.06GHz Xeon DP processor-based systems running Red Hat 9.0, with Myrinet 2000 interconnects, and an I/O subcluster with more than 120 terabytes of DataDirect storage. Tungsten provides Intel 8.0 icc and GNU gcc 3.2.2 for compilation, the Load Sharing Facility (LSF) batch system for job control and ChaMPIon/Pro for the MPI runtime environment.

The aforementioned methamphetamine case data was used to compare the two algorithms, LHOIM and HOIM. The sample dataset included 16 records containing 24 total items. Using this data, LHOIM generated 6809 latent itemsets while HOIM generated 4334 explicit itemsets based on 2045 link groups up to order ten. For the same input data, Apriori generated 75 itemsets including 24 1-itemsets. Table XI compares the number of itemsets generated at each level for Apriori, HOIM and LHOIM, while Table XII compares the average support at each level for the three algorithms.

TABLE XI.

NUMBER OF ITEMSETS GENERATED AT EACH LEVEL K

k	Apriori	HOIM	LHOIM
2	32	139	141
3	14	402	593
4	6	903	1830
5	N/A	1254	4245
6	N/A	1052	N/A
7	N/A	488	N/A
8	N/A	96	N/A

TABLE XII.

AVERAGE SUPPORT COUNT AT EACH LEVEL K

k	Apriori	HOIM	LHOIM
2	1.03125	0.512368	1.622969
3	1	1.192939	1.302798
4	1	1.883378	1.044705
5	N/A	2.584516	0.84219
6	N/A	3.217012	N/A
7	N/A	3.668064	N/A
8	N/A	3.861208	N/A

As depicted in Table XI, in Apriori the number of itemsets generated decreased after reaching a peak for lower levels of k. Also, in Table XII the support decreased with the increase in level. It is notable in Table XI that HOIM similarly sees decreasing numbers of itemsets after an early peak, although support for HOIM itemsets increased with increasing k (Table XII). In contrast, the number of latent itemsets discovered in LHOIM increased with increasing k (Table XI), but support decreased (Table XII). Because HOIM generates itemsets based on recordsets that are in turn formed in an Apriori-like manner, the number of itemsets generated using HOIM generally followed Apriori’s pattern. LHOIM, on the other hand, discovered itemsets based on link groups, which grow as the order increases. The longer the link groups, the larger the merged records as well as their corresponding size. Since LHOIM merged records contain more items, the level designating the peak of the number of itemsets increased as well. For example, for a total of 23 items, the maximum possible combinations of 2-itemsets is 253 while the potential maximum for 5-itemsets is 33649. This is one explanation as to why the number of itemsets generated by LHOIM continue to increase.

Another interesting pattern in Table XII demonstrates that support for HOIM tended to increase with the level, while the support for LHOIM tended to gently decrease. Increasing support in HOIM indicates that the sizes of recordsets continue to scale slightly faster than the order, despite the application of the square root and $\log_{10}$. This suggests that more experimentation is required to refine and explore suitable metrics. In LHOIM, the support stayed relatively stable, potentially due to repeatedly generating small itemsets across orders. In other words, for any link group, the sub-link groups are also used in LHOIM, the result of which is that the itemsets supported by the sub-link groups will be generated by all the super-link groups.

From this preliminary example, one can see that LHOIM and HOIM capture different characteristics of the data. For example, {"Del Green", "John Deer", "Smith Robert"} is a latent higher order itemset generated from the link *Del Green* $\sim^{\text{245 5th Memorial Dr. W.}}$ *Smith Robert* $\sim^{\textit{212-548-9638}}$ *John Deer* using LHOIM by linking records containing “Del Green.” On the other hand, {"Robert Gio", "Del Green", "John Davison", "Reu Robots"} is an explicit higher order itemset discovered by HOIM because the four records are 4th-order linked to each other. Although these four entities never appear together in any single link group, this higher order itemset suggests that they are closely coupled together.

In addition to empirical comparisons between the two algorithms, experiments evaluating the scalability of a higher order itemset generation algorithm were performed. These tests used the cluster at NCSA and scaled up to approximately 4,000 records on each of up to 128 processes on 64 nodes. The input again was the aforementioned methamphetamine case data. The output generated was higher-order itemsets. Correctness of program execution was determined by comparing the itemsets produced by each MPI process. In order to validate the system, the input data was designed such that each local MPI process produced identical itemsets. Although this is not the expected mode of operation, it serves to validate the system. Therefore, the algorithm was validated simply by recognizing identical local results across different nodes.

TABLE XIII.

PROCESSING TIMES WITH 4,096 RECORDS PER NODE

Processes (Nodes)	Total Records	User (sec)	System (sec)
2 (1)	8,192	0.64	0.14
4 (2)	16,384	0.53	0.14
8 (4)	32,768	1.1	0.39
16 (8)	65,536	1.59	0.39
32 (16)	131,072	3.64	1.0
64(32)	262,144	5.14	1.13
128(64)	524,288	14.33	3.1

The user and system execution times of the system are depicted in Table XIII. Higher-order itemsets generated by each MPI process were identical. Although these results are preliminary in nature, they serve to validate the higher order itemset mining algorithm.

X. CONCLUSIONS

This article presented the Distributed Higher Order Text Mining (D-HOTM) framework, a novel association rule mining algorithm that discovers propositional rules based on higher order associations in a distributed environment. D-HOTM provides a solution to distributed association rule mining of textual data from raw data pre-processing through post-processing analysis. The theoretical foundation of the mining and linking processes provided herein addresses many difficult challenges posed by heterogeneous distributed databases that cannot be easily centralized.

This work developed a framework to extend association rule mining from 1st-order to all possible orders. Definitions for higher order itemsets were given and the context for such itemsets defined. Two novel approaches to discovering higher order itemsets were presented and evaluated, including mechanisms to calculate support. A software system was designed, implemented and tested, and preliminary experimental results were presented and discussed. The results indicate that distributed higher order association rule mining provides valuable insight into item relations that is unavailable with traditional algorithms.

XI. Future Work

The future work of this project aims to extend every facet of the D-HOTM system. First, the system will be formally described and designed to serve as a computational service. Second, the object isomerism component will be extended to take into account more sophisticated metrics in determining object similarity. Third, to become a widely-usable and extensible service (by experts and novices alike), new metaphors, models, interaction and visualization techniques will be introduced across many aspects of this system. Finally, while the basic theoretical framework is presented here, much work is needed to enhance scalability, establish and verify metrics, etc.

The first objective is to provide the D-HOTM system as a resource to the network (for many data extractors and sharing services to use), through the use of open-source standards and software. Adopting the Service Oriented Architecture (SOA) approach will enable the system to be available as an on-demand service to the network, greatly simplifying adoption and usage by law enforcement personnel, systems and services.

The SOA model is a powerful choice for the enhanced D-HOTM system, as more of the link analysis can be rooted in distributed computational units, feeding the results to more lightweight clients as needed. Using this model, end-users only need platforms able to handle the visualizations and interface (which any current mid-grade computer system will provide). In addition, as data storage and transferal is based on NIEM and GJXDM, this system will be easily interconnected with other projects resulting from current and future DHS/DOJ initiatives. Scalability and stability are thus also key elements of any system that aims to provide a ubiquitous resource across the law enforcement data infrastructure.

A second direction of future work extrapolates the exploration of simple edit distance metrics for identifiers such as title concatenated with authors, and will continue to explore more sophisticated techniques based on a hybrid Fellegi-Sunter neural-network model currently under development in the TMI [44]. Evaluation of the system will be based on the standard techniques used in object isomerism research including, for example, the use of the F_β metric on testing data.

A third direction of future work deals specifically with the extensibility and applicability of the D-HOTM system, from the position of the end-users and developers. Issues of scalability, computational steering, interactive mining, distributed development standards, techniques and interfaces for different users will need to be addressed before data mining becomes widely utilized. With the D-HOTM system as a central functional core, various scaling and distribution mechanisms are being examined, including additional communication libraries (other than MPI), computing mechanisms (e.g., Condor and Xgrid), and Grid-enabling technologies (such as the

Globus toolkit). Moving data mining applications from the traditional batch-style to a more interactive experience is a related aspect of future work, as there are few systems that provide distributed monitoring and steering capabilities for mining systems. Visualizing and analyzing textual-based results is also challenging, as most visual analytics systems are heavily domain-dependent on Cartesian positional data (such as in the Biomedical and mechanical engineering domains). Even with solutions to these issues in place, much work remains to be done involving the number, kind, and definition of metrics for interactive, distributed, text mining systems.

Tackling these and other issues, while providing useful and effective solutions for users, is one of our long-term goals. Our present approach in addressing these questions is to provide a visual programming mechanism (facilitating component-oriented design) based on one or more description languages – languages that will provide a meta/semantic-level standard facilitating data transmission and transformation, software component requirements and products, message passing, metrics description and other aspects critical for large-scale distributed systems.

Fourthly and finally, in future work we plan to address both theoretical and practical aspects of these algorithms. First, we plan to apply these algorithms on real world data including e-Marketplace, law enforcement and public healthcare data as part of our ongoing NSF-funded work in distributed higher-order ARM development. In terms of algorithmic evaluation, we plan to develop methods for comparing high-confidence, high-support rules generated using both HOIM and LHOIM with the rules generated by Apriori. Also, as noted further work is needed in evaluating metrics for support. The current support metrics were developed empirically, and we need to further explore whether there is a theoretical basis for calculating support in this way, and also what other factors must be considered in calculating support.

D-HOTM has been developed using the Text Mining Infrastructure (TMI) [44], and along with the existing TMI release available at hddi.cse.lehigh.edu, we also plan to release D-HOTM as a platform for research and development of distributed text mining and association rule mining algorithms. Finally, we plan to address issues related to scalability of the algorithm. Although the current algorithm rests on a firm theoretical foundation, much of value remains to be accomplished in terms of improving the performance.

ACKNOWLEDGMENT

The authors wish to thank Lehigh University, the Pennsylvania State Police, the Lockheed-Martin Corporation, the City of Bethlehem Police Department, the National Science Foundation and the National Institute of Justice, US Department of Justice. This work was supported in part by NSF grant number 0534276 and NIJ grant numbers 2003-IJ-CX-K003, 2005-93045-PA-IJ and 2005-93046-PA-IJ. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of Lehigh University, the US Department of Justice, the National Science Foundation, the Pennsylvania State Police or the Lockheed Martin Corporation.

We are also grateful for the help of co-workers, family members and friends. Co-authors S. Li, C. D. Janneck, T. Wu, M. D. Dilsizian and W. M. Pottenger also gratefully acknowledge the continuing help of their Lord and Savior, Yeshua the Messiah (Jesus the Christ) in our lives and work. Amen.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," *Advances in knowledge discovery and data mining*, pages 307–328, 1996.

[3] S. Brin et al., "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *Proc. ACM SIGMOD Conf. Management of Data, ACM Press*, New York, pp. 255–264, 1997.

[4] J. S. Park, M. Chen and P. S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules," *Proc. ACM SIGMOD Conf., ACM Press*, New York, 1995, pp. 175–186.

[5] A. Savasere, E. Omiecinski and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," *Proc. 21st Int'l Conf. Very Large Databases,* Morgan Kaufmann, San Francisco, pp. 432–444, 1995.

[6] A. Mueller, "Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison," *Tech. Report CS-TR-3515,* Univ. of Maryland, College Park, MD, 1995.

[7] D. R. Swanson, "Complementary structures in disjoint science literatures," In A. Bookstein, Y. Chiaramella, G. Salton and V. V. Raghavan (Eds.), *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* pp. 280–289, New York: ACM Press, 1991.

[8] M. Ganiz, W. M. Pottenger and C. D. Janneck, "Recent Advances in Literature Based Discovery," *Journal of the American Society for Information Science and Technology, JASIST*, 2006 (submitted for publication).

[9] R. V. Hauck, H. Atabakhsh, P. Ongvasith, H. Gupta, H. Chen, "Using Coplink to analyze criminal-justice data," *IEEE Computer 35 (3),* 2002, pp. 30– 37.

[10] R. J. Mooney, P. Melville, L. R. Tang, J. Shavlik, I. C. Dutra, D. Page and V. S. Costa, "Relational Data Mining with Inductive Logic Programming for Link Discovery," *Proceedings of the National Science Foundation Workshop on Next Generation Data Mining*, Nov. 2002, Baltimore, MD.

[11] M. Ganiz, W. M. Pottenger and X. Yang, "Link Analysis of Higher-Order Paths in Supervised Learning Datasets," In *the Proceedings of the Workshop on Link Analysis, Counterterrorism and Security, 2006 SIAM Conference on Data Mining*, Bethesda, MD, April 2006.

[12] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient Similarity Search In Sequence Databases," In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, Springer Verlag, 1993.

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

[14] P. N. Tan, V. Kumar and J. Srivastava, "Indirect association: Mining higher order dependencies in data," *Technical Report TR00-037*, University of Minnesota, 2000.

[15] P. N. Tan and V. Kumar, "Mining Association Patterns in Web Usage Data," University of Minnesota, 2002.

[16] M. Spiliopoulou and J. F. Roddick, "Higher Order Mining: Modeling and Mining the Results of Knowledge Discovery," In *Data Mining II - Proc. Second International Conference on Data Mining Methods and Databases,* pp.309-320. Ebecken, N. and Brebbia, C.A. (eds). Cambridge, UK, WIT Press.

[17] S. Z. Li, T. Wu and W. M. Pottenger, "Distributed Higher Order Association Rule Mining Using Information Extracted from Textual Data," *SIGKDD Explorations*, Volume 7, Issue 1, June, 2005.

[18] L. Dehaspe and L. D. Raedt, "Mining association rules in multiple relations," In *ILP '97: Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 125–132, London, UK, Springer-Verlag,1997.

[19] S. Nijssen and J. Kok, "Faster Association Rules for Multiple Relations," In *IJCAI01*, Seattle, Washington, USA, pages 891-896, 2001.

[20] T. Wu and W.M. Pottenger, "A Semi-Supervised Active Learning Algorithm for Information Extraction from Textual Data," *JASIST*, volume 56, number 3, pages 258-271, 2005.

[21] H. Mannila, H. Toivonen and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, 259-289, 1997.

[22] K. Chan and A. Fu, "Efficient Time-Series Matching by Wavelets," In *Proc. of 1999 Int. Conf. on Data Engineering,* Sydney, Australia, March, 1999.

[23] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," In *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data*, Minneapolis, Minnesota, May, 1994.

[24] E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Springer-Verlag, Knowledge and Information Systems*, p. 263–286, 2001.

[25] H. Toroslu and M. Kantarcioglu, "Mining Cyclically Repeated Patterns," *Springer Lecture Notes in Computer Science* 2114, p. 83, 2001.

[26] J. Han, G. Dong, and Y. Yin, "Efficient mining partial periodic patterns in time series database," *Proc. ICDE*, 106-115, 1999.

[27] J. Yang, W. Wang and P. Yu, "Mining asynchronous periodic patterns in time series data," *Proc. SIGKDD*, 275-279, 2000.

[28] J. Yang, W. Wang and P. Yu, "Mining surprising periodic patterns," *Proc. SIGKDD*, 2001.

[29] W. G. Aref, M. G. Elfeky and A. K. Elmagarmid, "Incremental, Online and Merge Mining of Partial Periodic Patterns in Time-Series Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 3, pp. 332-342, 2004.

[30] C. Bettini, X. S. Wang, S. Jajodia and J. L. Lin, "Discovering frequent event patterns with multiple granularities in time sequences," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 10, iss. 2, Mar/Apr., pp. 222 –237, 1998.

[31] W. Pratt, and M. Yetisgen-Yildiz, "LitLinker: Capturing Connections across the Biomedical Literature," *Proceedings of the International Conference on Knowledge Capture (K-Cap'03),*. Florida, October 2003.

[32] P. Edmonds, "Choosing the word most typical in context using a lexical co-occurrence network," In *Proceedings of the Thirty-fifth Annual Meeting of the Association for Computational Linguistics*, pp. 507-509, 1997.

[33] X. Zhang, M. Berry, and P. Raghavan, "Level search schemes for information filtering and retrieval," *Information Processing and Management 37 (2),* pp. 313-334, 2000.

[34] H. Schütze, "Automatic Word Sense Discrimination," *Computational Linguistics* 24 (1), pp. 97-124, 1998.

[35] J. Xu and W. B. Croft, "Corpus-Based Stemming Using Co-Occurrence of Word Variants," *ACM Transactions on Information Systems* 16 (1), pp. 61-81, 1998.

[36] I. Fellegi and A. Sunter, "A theory for record linkage," *Journal of the American Statistical Association* 64, 328, 1183–1280, 1969.

[37] M. A. Jaro, "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," *Journal of the American Statistical Association*, 89:414-420, 1989.

[38] A. B. Tucker, "The Computer Science and Engineering Handbook," *CRC Press*, New York, 1996.

[39] W. E. Winkler, "Methods for Record Linkage and Bayesian Networks," *Research Report Series*, Statistics# 2002-05, 2002.

[40] W. Cohen and J. Richman, "Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration," In *SIGKDD*,2002.
[41] A. McCallum, K. Nigam and L. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," In *ACM SIGKDD,* pp. 169–178, 2000.
[42] Paul Embley and Robin Gibson, "History and Background," *Presented in the Global Justice XML Data Model Developer's Workshop*, Atlanta, Georgia, 2004.
[43] National Information Exchange Model webpage and tools, http://niem.gov/default.php and http://niem.gtri.gatech.edu/niem-ssgt/Introduction.do
[44] L. E. Holzman, T. A. Fisher, L. M. Galitsky, A. Kontostathis and W. M. Pottenger, "A Software Infrastructure for Research in Textual Data Mining*" The International Journal on Artificial Intelligence Tools*, volume 14, number 4, pages 829-849, 2004.
[45] M. Elfeky, V. Verykios, and A. Elmagarmid, "TAILOR: A Record Linkage Toolbox," In *Proceeding of the 18th Int. Conf. on Data Engineering*, IEEE, 2002.
[46] F. J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," *Communications of the ACM*, 7(3):171-176, 1964.
[47] V. I. Levenstein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Sov. Phys*. Dokl. 10:707-710, 1966.
[48] T. Uno, "An Output Linear Time Algorithm for Enumerating Chordless Cycles," *92nd SIGAL of Information Processing Society Japan*, 47-53, 2003.