A General Framework to Encode Heterogeneous Information Sources for Contextual Pattern Mining

Weishan Dong[†], Wei Fan[‡] [*], Lei Shi[♯], Changjin Zhou[†], and Xifeng Yan[§]
[†]IBM Research – China
[‡]Huawei Noah's Ark Lab
[♯]Institute of Software, Chinese Academy of Sciences
[§]University of California at Santa Barbara
{dongweis,zhoucj}@cn.ibm.com, wei.fan@gmail.com, shil@ios.ac.cn, xyan@cs.ucsb.edu

ABSTRACT

Traditional pattern mining methods usually work on single data sources. However, in practice, there are often multiple and heterogeneous information sources. They collectively provide contextual information not available in any single source alone describing the same set of objects, and are useful for discovering hidden *contextual patterns*. One important challenge is to provide a general methodology to mine contextual patterns easily and efficiently. In this paper, we propose a general framework to encode contextual information from multiple sources into a coherent representation—Contextual Information Graph (CIG). The complexity of the encoding scheme is linear in both time and space. More importantly, CIG can be handled by any single-source pattern mining algorithms that accept taxonomies without any modification. We demonstrate by three applications of the contextual association rule, sequence and graph mining, that contextual patterns providing rich and insightful knowledge can be easily discovered by the proposed framework. It enables Contextual Pattern Mining (CPM) by reusing single-source methods, and is easy to deploy and use in real-world systems.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Heterogeneous sources, contextual pattern mining

1. INTRODUCTION

Today's explosion in data does not only grow in the number of examples, but also grow in the number of available information sources describing related set of objects. Traditional frequent pattern mining methods for association rules [1, 8], sequences [2, 19], and graphs [20, 11] usually work on single information sources. The target objects of interest are simply a set of transactions, sequences composed of time-stamped elements, or labeled graphs. However, in reality, there are usually multiple heterogeneous information sources describing the contexts or related attributes of the target objects. These sources provide important contextual information not available from any single source alone, but can be useful to understand the data and discover interesting knowledge [13]. For example, in a crime scene analysis case, on Mondays, vehicle thefts happened more frequently in areas with high population density. Interestingly, the same type of crimes on Saturdays happened more frequently in low population density areas. Here the contextual information, i.e., the population density, from census data feeds but not available in the crime database, played an important role to extract useful crime patterns. Another example is that, in the epidemic spread characterization scenario, the location contexts such as the relative position to a river supplying drinking water, can be extracted from the map data source. Sequential patterns indicating that most people with diarrhea symptoms had visited downstream areas before getting sick can be discovered. A probable water-borne transmission and the source of contamination can therefore be inferred. Such insightful patterns cannot be found only by analyzing people's symptoms without utilizing the location contextual information of their movements. Most traditional data mining methods do not consider the contextual information from the heterogeneous sources, but focus on finding patterns explicitly appearing in a single source. Therefore, they are not directly applicable to these problems.

However, simply joining multiple data sources into a single dataset does not work either; it is well known that joining multiple sources not only results in an unnecessarily huge dataset but also leads to the loss of information at the same time (the semantics represented by the linkages between information sources) [5, 7]. Multi-relational data mining (MRDM) [4, 5] can be applied to mine multiple sources. However, in a real-world system, such as a modern enterprise's data analytics platform or a commercial business intelligence product, it is usually expensive to deploy a new set of algorithms, as it often involves intensive testing and requires a lot of engineering effort. Yet, another practitioners' concern recently raised from real-world applications of mining multiple sources is the "ease of use" [14].

[*]Part of this work was done when the author was in IBM T.J.Watson Research Center.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

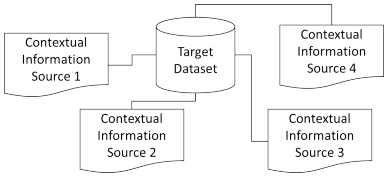


Figure 1: Star Schema of Information Sources

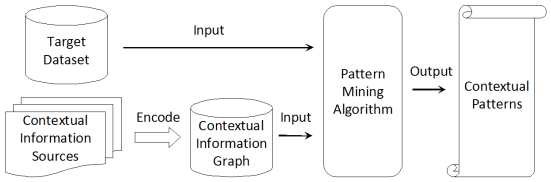


Figure 2: Contextual Pattern Mining

In this paper, we answer the question that: given a set of traditional single-source pattern mining algorithms, how to non-intrusively and maximally reuse them for discovering insightful patterns across multiple heterogeneous sources? In real-world applications, one of the most common ways in which multiple information sources relate to each other is the star schema, which consists of a target dataset and a number of heterogeneous contextual information sources. The target dataset is a main information source of objects that are of primary interest, where the primary key identifies the *target objects* (a target object can be a transaction, a sequence, or a graph in the dataset), and several foreign keys identify the *identities* inside a target object (an identity can be an item inside a transaction or sequence, or a label of graph vertex or edge). The contextual information sources provide auxiliary information about all or a subset of objects in the target dataset, where the primary key is linked to a foreign key (possibly via predicates, Section 2.3) in the target dataset. These information sources contain the features (either constant or dynamic over time) of the identities inside the target objects, which provides rich context for mining insightful patterns of target objects. Figure 1 is an example of star schema of a target dataset and four contextual information sources. We define patterns discovered from such multiple sources as *contextual patterns*:

Definition 1 (Contextual Pattern). *A pattern of the target objects, which contains the information explicitly appearing in the target dataset and the information implicitly represented across the contextual information sources.*

Apparently, contextual patterns can only be mined by collectively considering the target dataset and the contextual information sources. Figure 2 illustrates the flow of the proposed Contextual Pattern Mining (CPM) concept. As shown, the idea of CPM is quite different from the traditional MRDM approaches, although sharing the same objective. By solving the problem from a new perspective, CPM outperforms the traditional MRDM approaches, in terms of non-intrusiveness, ease of use, and efficiency. More details can be found in Section 6. The main challenge of CPM is how to effectively encode multiple sources of contextual information before running a traditional single-source pattern mining method, so that contextual patterns can be easily discovered. In this paper, we propose the Contextual Information Graph (CIG) encoding framework to solve this problem. An example of CIG in the crime analysis scenario is given in Figure 3. It encodes the population densities (POPDEN) as well as the ratios of male to female (RMF) of census tracts where the crime had happened. With CIG, a

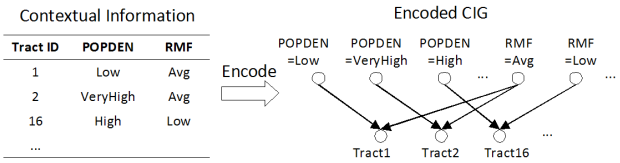


Figure 3: Contextual Information Graph (CIG)

single-source pattern mining algorithm accepting taxonomy as an additional input can be applied to the target crime history dataset. We also show that simply joining contextual information to the target dataset does not work, and can result in a large number of redundant patterns (Section 5). The advantages of the proposed CIG encoding framework include: (1) Generality. CIG supports mining all the three types of popular frequent patterns: itemset/association rule, sequence, and graph. (2) Efficiency and scalability. With the HashSet and HashMap data structure, both the time and space complexities of CIG encoding are linear. The Knowledge-to-Identity path query on CIG in pattern mining stage (Section 2.2) is constant in time. (3) Robustness. CIG is capable of handling missing values, dynamically changing values, and multiple values on contextual features. (4) No information loss. CIG encoding is a one-to-one mapping of the contextual information sources.

2. ENCODING FRAMEWORK

In most cases, frequent pattern mining deals with discrete features [7]. In this paper, we assume all the contextual information (including time information) is in discrete feature values, or has been discretized according to some criterions. To effectively conduct Contextual Pattern Mining (CPM), we encode all available contextual information by a CIG, which is a labeled DAG (Directed Acyclic Graph) with labels associated only with its vertices. Let U denote the set of contextual information, V denote the set of identities, and E denote the set of linkages between U and V.

Definition 2 (Contextual Information Graph). *A Contextual Information Graph (CIG) is a labeled bipartite directed graph $G = (U, V, E)$, s.t. (1) $\forall u, v \in U \cup V, \ell(u) \neq \ell(v)$ if $u \neq v$, and (2) $\forall$ edge $(u, v) \in E, u \in U, v \in V$, where ℓ is the label function of the vertices in G.*

An example is given in Figure 3. Available contextual information collected from census data feeds consists of two features (POPDEN and RMF) for each identity (census tract). Here U is the set of all possible combinations of a feature and its value (vertices in the upper half of the CIG), where one vertex stands for a combination. V is the set of all identities (vertices in the lower half of the CIG), where one vertex stands for an identity. The meaning of a vertex is shown by its label displayed aside. The directed edges from a vertex in U to a vertex in V denote the linkages between the identities and the feature values, which constitute set E. As can be seen, the first condition in Definition 2 means that in a CIG $G = (U, V, E)$, the label of a vertex is unique that only one vertex in U stands for a specific feature value and only one vertex in V stands for a specific identity. The second condition means that the linkages exist only between the identities and the feature values, and are represented by directed edges from a feature value to an identity.

2.1 Encoding Algorithm

Algorithm 1 summarizes the CIG encoding procedure. For some applications, contextual information is not fixed, and

Algorithm 1 CIG Encoding

Input: A set of contextual information sources $\mathcal{S}$.
Output: A CIG $G = (U, V, E)$.
1: $U \leftarrow \emptyset; V \leftarrow \emptyset; E \leftarrow \emptyset$;
2: **foreach** contextual information source $s \in \mathcal{S}$
3: **foreach** state of identity i at time t, $i_t \in s$
4: insert a vertex v to V, $\ell(v) \leftarrow i_t$;
5: **foreach** combination of feature F and its value X, $(F = X) \in i_t$
6: vertex $u \leftarrow$ search in U with $\ell(u) = (F = X)$;
7: **if** u is not found **then**
8: insert a vertex u to U, $\ell(u) \leftarrow (F = X)$;
9: insert a directed edge (u, v) to E;
10: **return** $G = (U, V, E)$;

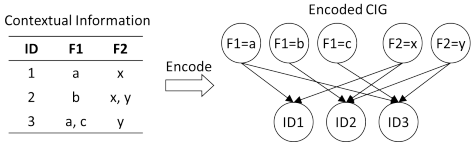


Figure 4: CIG of Fixed Contextual Information

can be changing over time. For example, temperature of an outdoor location varies over time, and traffic flow of a road may change between working hours and midnight. Therefore, on line 3, if the features of an identity change over time, a vertex indicating the state at a specific discrete time frame will be created. In reality, multiple values for one feature can exist. In this case, the algorithm reads every possible combination of the feature and its values, and creates separate vertices. Figure 4 is an example of encoding *fixed* contextual information from one source into a CIG. ID indicates the identities. F1 and F2 are two multi-valued features. The labels of vertices are shown inside. Figure 5 encodes *dynamic* contextual information from one source. The only difference in the CIG is that an identity state vertex is defined by its ID together with a time stamp. When encoding multiple sources, the resulting CIG is the union of several small CIGs, each is constructed from one source.

2.2 Contextual Pattern Mining

As shown in Figure 2, we propose to perform Contextual Pattern Mining in two steps: (1) construct a CIG from the contextual information sources, and (2) apply a single-source pattern mining algorithm to discover contextual patterns. The single-source pattern mining algorithm can be implemented based on an existing Generalized Pattern Mining (GPM) algorithm [16, 6, 17, 9, 3] with only minor modification or even no change at all. GPM algorithms accept a taxonomy of the identities besides reading a target dataset as primary input, and can mine generalized patterns across different levels of the taxonomy. A taxonomy describes the concept hierarchies (ontologies) of identities, typically appearing as a tree with great depth or forest structures. In most cases, a taxonomy is defined by human knowledge and constructed manually. Figure 6 is an example of a taxonomy of items. In a taxonomy, all the leaf nodes are specific identities appearing in target objects. The non-leaf nodes are virtual concepts defined by human knowledge. A directed edge from a higher level concept to a lower level concept, or from a concept to an identity indicates the conceptual generalization, i.e., reversely *is-a* semantics. For instance, in Figure 6, concept *Outerwear* generalizes identities *Jacket* and *Ski Pants*, thus *Jacket is-a Outerwear*, and *Ski Pants is-a Outerwear*. With this taxonomy, generalized association rule mining [16, 6] can discover rules like "*Outerwear* $\Rightarrow$ *Hiking Boots*" from a

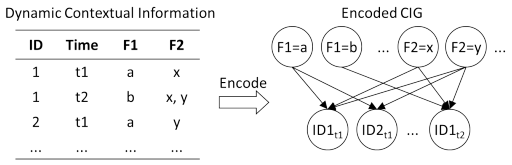


Figure 5: CIG of Dynamic Contextual Information

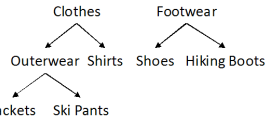


Figure 6: Taxonomy of Items (adopted from [16])

dataset of item purchase transactions, meaning that the customers bought *Outerwear* also bought *Hiking Boots*. However, specialized rules only at identity (item) level, such as "*Jacket* $\Rightarrow$ *Hiking Boots*" and "*Ski Pants* $\Rightarrow$ *Hiking Boots*", may not be frequent or confident. Generalized sequential pattern mining [17] and generalized graph mining [9, 3] work in similar ways.

CIG is different from taxonomy in the following aspects: (1) A CIG can be automatically constructed with Algorithm 1, whereas defining a taxonomy heavily requires involving human knowledge in most cases. (2) In a CIG $G = (U, V, E)$, there is no edge connecting two vertices in U, i.e., no linkage exists between the vertices representing contextual information. However, in a taxonomy, such linkages define relationships between two concepts. (3) In a typical taxonomy of a tree (or forest) structure, vertices representing identities have in-degree = 1. Whereas in a CIG $G = (U, V, E)$, vertices in V representing identities have $1 \leq$ in-degree $\leq |U|$. (4) We can regard both contextual information and concepts as knowledge. It can be proven that if a path exists in a CIG that starts from a vertex representing knowledge and ends at a vertex representing an identity, it always contains only one edge. But in a taxonomy, such paths can contain multiple edges. Generally, we call such a path the Knowledge-to-Identity (KI) path.

Definition 3 (Knowledge-to-Identity (KI) Path). *Given a CIG $G = (U, V, E)$, a KI path exists from knowledge $(F = X)$ to identity i iff. $\exists (u, v) \in E$, s.t. $\ell(u) = (F = X), \ell(v) = i, u \in U, v \in V$. We denote it by $(F = X) \triangleright i$.*

In existing GPM algorithms, the KI path is usually referred as the antecedent-descendant relationship on a taxonomy. In most cases, the KI path query is the only necessary interface that a GPM algorithm interacts with a taxonomy [16, 6, 17, 9, 3]. To be specific, a GPM algorithm only needs to know the followings to mine generalized patterns, all of which are essentially KI path queries: (1) Given an identity, identify all the antecedents, i.e., concepts generalizing the identity. (2) Given a concept, identify all the descendants, i.e., lower level concepts and identities specializing the concept. (3) Given a concept and an identity, check if there exits a path connecting them. We can see that as long as these KI path queries can be done on CIG, a GPM algorithm can accept a CIG by blindly treating it as a taxonomy, and the output will be contextual patterns.

This is a favorable characteristic: We can conduct CPM by non-intrusively leveraging traditional GPM algorithms, with an appropriate way of encoding new information that GPM algorithms cannot directly handle. And indeed, performing the KI path queries on CIG is straightforward (Section 2.4). We can find that CPM can be done with a rela-

tively small cost. With the CIG encoding, existing single-source GPM algorithms can be reused to the maximum level. One thing should be noticed is that, if the GPM algorithm applied only accepts tree (or forest) structured taxonomies, it may need minor changes so that a general DAG structured CIG can be processed. In this case, the time complexity bounds of a GPM algorithm [16, 6, 17, 9, 3] does not change if replacing a taxonomy input by a CIG. This is because in a general sense, KI path query on either a taxonomy or a CIG shares the same time complexity if the taxonomy and the CIG are isomorphic. Besides, since CIG is a bipartite graph, KI path queries on it can be highly efficient (Section 2.4). It is also feasible to combine a taxonomy to a CIG, so that the algorithm can discover contextual patterns and traditional generalized patterns simultaneously. However in this case, KI path queries are essentially done on a DAG not necessarily being bipartite. Such KI path queries become the general graph reachability queries. Labeling and indexing methods [18] can be employed to accelerate the queries. Specific CPM algorithms for mining contextual association rules, sequences, and graphs will be presented in Section 4.

2.3 Indirect Contextual Information

In some cases, contextual information may not be directly linked to identities, but via *predicates*. Take the crime scene analysis as example. To study the factors contributing to crime, contextual information such as location features, need to be considered. The location of a crime is usually represented by GPS coordinate in IT systems of police departments. However, environmental contexts, such as population densities of the communities and types of the buildings nearby, do not directly relate to the GPS location. In this case, a *predicate* defining a relationship between two identities is needed to correlate available information. Specifically, in the crime example, spatial predicates (e.g., *within*, *close_to*) [10] can define spatial relationships between a GPS location identity and the contextual identities (community, building, etc.). The computation of predicates is usually done externally (e.g., by a computation engine or a lookup table), but a predicate itself can still be directly represented in the target dataset. Table 1 illustrates an example of a target dataset relating to contextual identities via predicates. Consider each target object has a feature F1. The values of F1 include predicates $p_1(\cdot)$ and $p_2(\cdot)$. Using the crime example, F1 could denote the spatial features derived from GPS coordinates, and p_1 and p_2 may stand for *within* and *close_to* relationships, respectively. ID1 and ID2 that relate to target objects via p_1 and p_2 could stand for a community and a building, respectively. CPM treats an instantiated predicate (in the above example, p_1(ID1) or p_2(ID2)) as a whole to be one item. It works by simply defining the following.

DEFINITION 4. *Given a CIG G and a predicate $p(\cdot)$, $p(F = X) \rhd p(i)$ on G iff. $(F = X) \rhd i$ on G.*

For example, given the CIG in Figure 4 and the dataset in Table 1, we have p_1(F1=a) $\rhd$ p_1(ID1), p_1(F2=x) $\rhd$ p_1(ID1), p_2(F1=b) $\rhd$ p_2(ID2), and p_2(F2=x) $\rhd$ p_2(ID2). By doing so, CPM can deal with predicates as well as ordinary identities.

2.4 Implementation & Complexity Analyses

In our implementation, a HashSet is used to store the vertices $U \cup V$ given a CIG $G = (U, V, E)$. One HashMap is used to store the directed edges E, where each key is a vertex $u \in U$, and its mapped value is a HashSet of all its descendant vertices on G, defined by $\{v | v \in V, (u, v) \in E\}$. Another HashMap representing the reverse edges is also maintained, where each key is a vertex $v \in V$, and its mapped value is a HashSet of all its antecedents, defined by $\{u | u \in U, (u, v) \in E\}$. These two HashMaps are built simultaneously on line 9 in Algorithm 1.

Table 1: Dataset Relating Contexts via Predicates

	F1
TargetObj1	p_1(ID1)
TargetObj2	p_1(ID1), p_2(ID2)
...	...

Time Complexities: On HashSet/HashMap, it is known that any insertion or search operation is $O(1)$ asymptotically, assuming there is no collision. In practice, this is usually true, especially when sufficiently large capacity of the HashSet/HashMap is available as compared to the total number of elements (keys) to be stored. In this case, in Algorithm 1, line 4 (vertex insertion), line 6 (vertex search) and line 9 (edge insertion) are all constant in time. The time complexity of Algorithm 1 is thus $O(\sum_{\forall s \in S} \sum_{\forall i_t \in s} \sum_{\forall (F=X) \in i_t} 1) = O(|E|)$, linear to the amount of information encoded[1]. The worst case of HashSet/HashMap operations is $O(n)$ only if, in the extreme case, that all hashes collide for the total n elements in the HashSet/HashMap. Figure 7 demonstrates a simple example of Hash collision. Therefore, in most cases, the time complexity of CIG encoding is linear. Only once scanning of all the contextual information is sufficient to construct CIG. Because CIG is always a bipartite graph, KI path query is equivalent to edge search. Given a constructed CIG, assuming no collision, searching all descendants or antecedents of a vertex and searching an edge defined by two specified vertices in the HashMaps, are all constant in time. Similarly, the worst cases are $O(|U|)$, $O(|V|)$, and $O(|U|+d)$, respectively, where d is the max degree of vertices in U. In a word, any KI path query on CIG is constant in time in most cases, where there is no hash collision.

Space Complexities: A HashSet/HashMap typically has space complexity of $O(n)$, where n is the number of elements stored. In our case, the HashSet for vertices has $O(|U \cup V|) = O(|U| + |V|)$ space complexity. We can also infer that the two HashMaps have $O(|U|+|E|)$ and $O(|V|+|E|)$ space complexities, respectively. Therefore, the overall space complexity of a CIG is $O(|U| + |V| + |E|)$.

3. EXPERIMENTS

In this section, we experimentally validate the linear time and space complexities of CIG encoding and the constant time complexity of KI path query. All the experiments are done on a machine with Intel Core2 2.66GHz CPU and 2GB memory, using Java programming with the initial capacity of 16 and the load factor of 0.75 for HashSet/HashMap. The results are shown in Table 2 and Figure 8, which are obtained on the contextual information used in Section 4.

[1]When the information source is a typical flat table of M rows and N columns, this complexity is $O(M \cdot N)$, and $|E| = M \cdot N$.

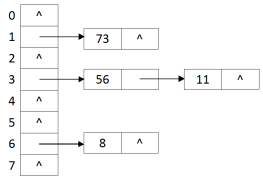
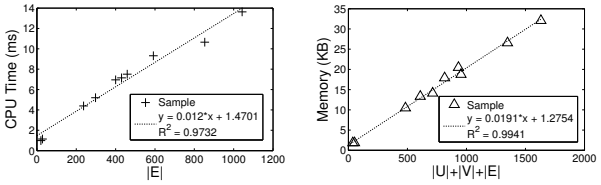


Figure 7: HashSet of Capacity 8. Keys $\{8, 56, 11, 73\}$ stored with hash function x mod 8. 56 and 11 collide.

Table 2: Results of Time and Space Complexities

$\|U\|$	$\|V\|$	$\|E\|$	CIG encoding CPU time (ms)	CIG memory usage (KB)	KI path query CPU time (ms)*
7	10	20	0.99	1.832	23.82
10	14	28	1.14	1.844	23.86
9	237	237	4.39	10.465	17.11
13	298	298	5.20	13.324	23.40
15	400	400	6.95	17.9	16.26
15	269	430	7.15	14.116	18.71
14	459	459	7.51	20.472	17.19
15	347	592	9.32	18.758	26.16
16	479	853	10.65	26.56	21.27
16	572	1043	13.63	32.082	19.31

*Total CPU time of 10,000 random KI path queries.



(a) Time Complexity (b) Space Complexity

Figure 8: Linear Complexities of CIG Encoding

The contextual data is represented by the integer type to eliminate the effects of varying hash code calculation time and storage size caused by data types like string. We can find that the CPU time of CIG encoding is linear to $|E|$, and the storage (space) of CIG is linear to $(|U|+|V|+|E|)$. Two linear equations fitting the samples are also shown in Figure 8. Both have the R square highly close to 1, indicating the linearities. To verify the constant time complexity of KI path queries, in each test, we randomly generate KI path queries 10^4 times and report the total CPU time. It is shown from Table 2 that the KI path query is efficient and only costs nearly constant time in the order of 20ms across all tests, in spite of the data change. In summary, all the results are consistent with the analyses in Section 2.4.

4. CASE STUDIES

Applications of specific contextual pattern mining algorithms to three case studies are given in this section.

4.1 Contextual Association Rule Mining

Data Overview: The first case study is on the real crime history of the City of Spokane, WA, USA. Contextual association rule mining is conducted to find crime patterns. Totally 816 crime incidents of 3 general types, *Drugs* (167 incidents), *Vehicle Theft* (552 incidents), and *Robbery* (97 incidents), reported in the northeast city (covering 10 census tracts) during Jan 2009 to Mar 2010, constitute the target dataset. The day of week on which the incident happened and its general type are included in pattern mining as normal attributes. The geometries of the census tracts and 23 major streets in the area are also used to derive spatial features for each incident, including *within* a tract and *close_to* a street (distance < 500 feet). A sample of the target dataset in transaction format is given in Table 3 (tid indicates id of transaction/incident). Figure 9 is a map overview of the data. The US Census Bureau data is involved as a contextual information source. Population density (per sq mi) and ratio of male to female are considered as contexts. Because these two features, according to the domain knowledge, can influence the distribution of crime occurrences. The tracts' features are summarized in Table 4. A CIG that encodes the POPDEN and RMF is constructed for CPM. A portion of the CIG is shown in Figure 3. A simple taxonomy of "every street *is-a Road*" is also combined as part of the CIG.

Table 3: Sample Crime Target Dataset

tid	items
4	Wednesday, Drugs, within(Tract26), close_to(Trent Av)
251	Thursday, Robbery, within(Tract16), close_to(Garland Av)
266	Friday, Vehicle Theft, within(Tract1), close_to(Freya St), close_to(Wellesley Av)

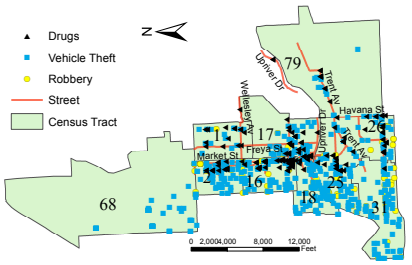


Figure 9: Map Overview of Crime Dataset

Contextual Association Rule Mining: The contextual association rule mining is summarized in Algorithm 2. As in the generalized association rule mining [16], first the transactions are expanded with the contextual information related to items using KI path queries on the CIG, where an item can be a predicate or an identity. Duplicate features expanded, if any, are removed. For example, after expansion, Table 3 will become Table 5. Then frequent itemsets are enumerated and pruned as in typical generalized association rule mining. Any itemsets containing an item $\triangleright$ the other are pruned to prevent generating itemsets like {within(Tract26), within(POPDEN=Low)} and {close_to(Trent Av), close_to(Road)}. Such itemsets indicate the correlation between an item and its context, which can be frequent but provide redundant knowledge already represented in the CIG. It can be proven that only pruning 2-itemsets is sufficient (Lemmas 1 and 2 in [16]) to eliminate such redundant patterns.

Algorithm 2 Contextual Association Rule Mining

Input: Target transactions $\mathcal{D}$; CIG G; Minimum support (min_sup); Minimum confidence (min_conf).
Output: Contextual association rule set $\mathcal{C}$.
1: **foreach** item i in each row of $\mathcal{D}$
2: expand the row with all the contexts $\triangleright i$ on G, removing any duplicates, to construct $\mathcal{D}'$;
3: find all candidate 2-itemsets from $\mathcal{D}'$;
4: **foreach** candidate 2-itemset $\{i_1, i_2\}$
5: **if** $i_1 \triangleright i_2$ **or** $i_2 \triangleright i_1$ **then** prune this 2-itemset;
7: continue to find all frequent itemsets using min_sup ;
8: $C \leftarrow$ find all confident association rules using min_conf ;
9: **return** C;

Results: We use min_sup = 2% and min_conf = 80%. No pattern about *Robbery* can be found, which indicates the randomness of these crimes. If only the crime target dataset is regarded as input without other information, only 12 rules including the following two can be found (support and confidence are shown in brackets, $\wedge$ indicates "AND"):

within(Tract18) $\wedge$ close_to(Upriver Dr) $\Rightarrow$ Vehicle Theft (2.2%, 90%)
within(Tract68) $\Rightarrow$ Vehicle Theft (6.4%, 88.1%)

If the taxonomy of streets is further involved, 20 rules (including the above 12) can be found, corresponding to the output of typical generalized association rule mining. Below list two new rules. Both can be verified in Figure 9.

within(Tract79) $\Rightarrow$ close_to(Road) (2.3%, 86.4%)
Drugs $\Rightarrow$ close_to(Road) (16.5%, 80.8%)

If the CIG encoding POPDEN and RMF features is further involved for CPM, 212 rules (including the above 20) can be found, including the followings.

Table 4: Features of Census Tracts

ID	POPDENSITY*	RATIO_MF*	POPDEN	RMF
1	954.447	1.02	Low	Avg
2	6381.73	0.96	VeryHigh	Avg
16	4715.44	0.88	High	Low
17	2879.07	0.96	Avg	Avg
18	4426.01	0.88	High	Low
25	6027.64	0.98	VeryHigh	Avg
26	569.551	1.10	Low	High
31	1060.55	1.17	Low	High
68	717.115	0.95	Low	Avg
79	543.256	1.10	Low	High

*Original numerical data. Discretized POPDEN & RMF are used for CPM.

Table 5: Samples of Expanded Crime Target Dataset

tid	items
4	Wednesday, Drugs, within(Tract26), close_to(Trent Av), within(RMF=High), within(POPDEN=Low), close_to(Road)
251	Thursday, Robbery, within(Tract16), close_to(Garland Av), within(RMF=Low), within(POPDEN=High), close_to(Road)
266	Friday, Vehicle Theft, within(Tract1), close_to(Freya St), close_to(Wellesley Av), within(RMF=Avg), within(POPDEN=Low), close_to(Road)

Saturday $\wedge$ within(POPDEN=Low) $\wedge$ within(RMF=High)
$\Rightarrow$ Vehicle Theft $(3.7\%, 81.1\%)$

Monday $\wedge$ within(POPDEN=VeryHigh) $\Rightarrow$ Vehicle Theft $(2.5\%, 80\%)$

Drugs $\wedge$ within(RMF=High) $\Rightarrow$ close_to(Road) $(4.3\%, 83.3\%)$

Drugs $\wedge$ within(RMF=Low) $\Rightarrow$ close_to(Road) $(5.3\%, 89.6\%)$

These results show that, on Saturdays, *Vehicle Theft* crimes are more probable to occur in tracts with POPDEN=Low and RMF=High. However, on Mondays, tracts with POPDEN=VeryHigh have more chance to report such type of crimes. For *Drugs* crimes, RMF=High or Low can increase the probability of reporting the incidents near a road. Meanwhile, no pattern of *Drugs* and RMF=Avg can be discovered. The detected rules reveal the hidden patterns of the crimes and provide extra insights.

4.2 Contextual Sequential Pattern Mining

Data Overview: The dataset is from the IEEE VAST Challenge 2011 Mini-Challenge 1. It contains 1,023,077 microblogs collected during Apr 30–May 20, 2011 (21 days), each has GPS coordinates, date, and author id. With text analysis, each microblog is tagged with 34 candidate epidemic keywords as its features, include: fever, diarrhea, cough, ache, etc. Among all, two keywords, fever and diarrhea, are used to derive two sequence datasets. The fever dataset consists of all microblogs from authors who sent at least one message tagged by *fever*. The diarrhea dataset consists of all microblogs from authors who sent at least one message tagged by *diarrhea*. There are 40,472 microblogs sent by 2,786 authors in the fever dataset, and 16,353 microblogs sent by 1,196 authors in the diarrhea dataset. For each author, a sequence of microblogs s/he sent is constructed. Each microblog is an element (itemset) of a sequence. The statistics of lengths and number of distinct dates of sequences in the two datasets are summarized in Table 6. It can be seen that in both datasets, on average, an author sent at least one microblog every two days. We can say that the sequences reflect the general movements of the people. Studying these sequences can help conduct some reasonable characterization on the spreads of fever and diarrhea.

Figure 11 shows the city map, which serves as the contextual information source for this case study. There are 13 districts, 6 water bodies (a river and 5 lakes) and 13 hospitals in the city. The river flowing from north to southwest, is known as part of the drinking water supply system. Using the map, the spatial features *within*$(\cdot)$ and *close_to*$(\cdot)$ (distance $<$ 1km) are derived for microblogs. A sample sequence in the fever dataset is shown in Tables 7. Sequences in the diarrhea dataset are similar and omitted. Importantly,

Table 6: Statistics of Sequences in The Two Datasets

Sequence Length	Fever	Diarrhea	*#Distinct Date*	Fever	Diarrhea
Range	[10, 18]	[9, 15]	Range	[7, 15]	[6, 14]
Mean	14.527	13.673	Mean	10.528	10.299

Table 7: A Sample Sequence in Fever Dataset

author	date	items
42	2011-4-30	within(Lakeside)
42	2011-5-3	within(Uptown)
	...	...
42	2011-5-18	within(Downtown), close_to(Vastopolis City Hospital)
42	2011-5-19	fever, within(Eastside)
	...	...

features of the districts including their relative orientations to the river (up/mid/downstream) and to the entire region (north/south/east/west/center) (Table 8), are involved as contextual information and encoded by a CIG (Figure 10). A taxonomy indicating that "every hospital *is-a Hospital*" is also combined as part of the CIG.

Contextual Sequential Pattern Mining: The contextual sequential pattern mining is summarized in Algorithm 3. With the same CIG, we apply Algorithm 3 once to the fever dataset and once to the diarrhea dataset, to find unique patterns for each. Similar to the contextual association rule mining, first the sequences are expanded via KI path queries on the CIG. Redundant patterns here refer to those sequences with an element (itemset) containing both an item and its contexts. Such an element rephrases the linkage information already represented the CIG, but provides nothing useful for new pattern discovery. Patterns with such elements are pruned during mining the sequences. Three additional time constraints are implemented: window-size, min-gap, and max-gap [16]. Explicitly, we set windows-size = 0 day, enforcing that all the items in an element must occur within the same day, but not necessarily from the same microblog. Min-gap and max-gap determine the minimum and the maximum time difference between two consecutive elements, respectively. We set min-gap = 0 day. Considering the frequency of people sending microblogs and the intuition that short term activities affect people's health more, we set max-gap = 2 days. It indicates that an element can occur at most 2 days before the next element. The dates of microblogs are also regarded as items in pattern mining.

Algorithm 3 Contextual Sequential Pattern Mining

Input: Target sequences $\mathcal{D}$; CIG G; min_sup .
Output: Contextual sequential pattern set $\mathcal{C}$.
1: **foreach** item i in each element of each sequence in $\mathcal{D}$
2: expand the element with all the contexts $\triangleright i$ on G, removing any duplicates, to construct $\mathcal{D}'$;
3: $C \leftarrow$ find frequent sequences from $\mathcal{D}'$ using min_sup , pruning any sequences with an element containing items i_1 and i_2, s.t. $i_1 \triangleright i_2$ **or** $i_2 \triangleright i_1$;
4: **return** C;

Results on Fever Dataset: We use min_sup = 40%. Since we are interested in knowing where the people visited before they mentioned fever, sequences ending with *fever* are selected and analyzed. Without the contextual information, traditional sequence mining only finds four patterns as below (support shown in brackets):

$$\text{2011-05-18} \wedge \text{fever} \ (49.0\%) \quad (1)$$

$$\text{2011-05-19} \wedge \text{fever} \ (44.1\%) \quad (2)$$

$$\text{2011-05-18} \rightarrow \text{fever} \ (51.9\%) \quad (3)$$

$$\text{within(Downtown)} \rightarrow \text{fever} \ (48.0\%) \quad (4)$$

Patterns (1) and (2) are sequences of only one element (a 2-itemset). Patterns (1)–(3) indicate that people mentioned fever mostly on May 18 and 19. These imply the fever outbreak may start from May 18. Pattern (4) implies that the

Table 8: Features of Districts

District ID	Relative Orientation (RO)
Villa	north, west, south
Westside	downstream, north, west, south
Riverside	upstream, north
Suburbia	north, east
Smogtown	downstream, south, west
Plainville	downstream, midstream, south, west
Downtown	midstream, center
Uptown	upstream, center
Southville	south
Lakeside	south, east
Cornertown	north, west
Northville	midstream, north
Eastside	east

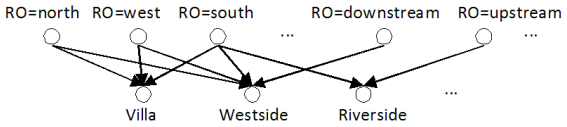


Figure 10: CIG Encoding Contexts from Table 8

ground zero location for fever could be Downtown because people mentioned fever just after they visited Downtown in the last two days. However, as shown below, this is far from the whole picture. CPM with the contextual information of districts' features finds four additional patterns ("RO=" omitted in *within*(·)):

$$\text{within(midstream)} \rightarrow \text{fever} \quad (55.5\%) \tag{5}$$

$$\text{within(center)} \rightarrow \text{fever} \quad (62.6\%) \tag{6}$$

$$\text{2011-05-18} \wedge \text{within(center)} \rightarrow \text{fever} \quad (41.2\%) \tag{7}$$

$$\text{within(midstream)} \wedge \text{within(center)} \rightarrow \text{fever} \quad (49.1\%) \tag{8}$$

These imply that not just the people who visited Downtown, but more precisely, the ones who visited the central and midstream areas would mention fever. CPM also finds patterns that start with *fever* and end with *close_to(Hospital)*:

$$\text{fever} \rightarrow \text{close\_to(Hospital)} \quad (47.2\%) \tag{9}$$

$$\text{fever} \rightarrow \text{2011-05-20} \wedge \text{close\_to(Hospital)} \quad (41.1\%) \tag{10}$$

$$\text{fever} \wedge \text{within(center)} \rightarrow \text{close\_to(Hospital)} \quad (40.0\%) \tag{11}$$

It is revealed that, after the people got fever (especially in the city center), they went to hospitals on May 20. These additional insights can be vital for government and medical agency. They could collect relevant information from the hospitals to understand the medical nature of the epidemic, and take actions to control the disease spread.

Results on Diarrhea Dataset: We use a larger min_sup = 80%, as it is observed that people's movements in this dataset are not as random as in the fever dataset. Similarly, sequences ending with *diarrhea* are analyzed. Traditional sequence mining without the contextual information cannot find any pattern. In contrast, CPM finds the followings (only closed patterns [19] are shown here to save space):

$$\text{within(west)} \wedge \text{within(south)} \rightarrow \text{diarrhea} \quad (88.0\%) \tag{12}$$

$$\text{2011-05-19} \wedge \text{within(west)} \wedge \text{within(south)} \rightarrow \text{diarrhea} \quad (85.8\%) \tag{13}$$

$$\text{within(downstream)} \wedge \text{within(west)} \wedge \text{within(south)} \rightarrow \text{diarrhea} \quad (83.9\%) \tag{14}$$

$$\text{2011-05-19} \wedge \text{within(downstream)} \wedge \text{within(west)} \wedge \text{within(south)} \rightarrow \text{diarrhea} \quad (82.0\%) \tag{15}$$

These imply that people mentioned diarrhea mostly after visiting the southwest and downstream areas on May 19. If instead using min_sup = 40% as on the fever dataset, a pattern indicating the diarrhea outbreak started and lasted after May 18, i.e., on May 19–20, can also be found:

$$\text{2011-05-18} \rightarrow \text{diarrhea} \quad (44.1\%) \tag{16}$$

However, still no pattern like "diarrhea → close_to(Hospital)" is detected, suggesting the patients with diarrhea did not go to hospitals for treatment although they felt sick.

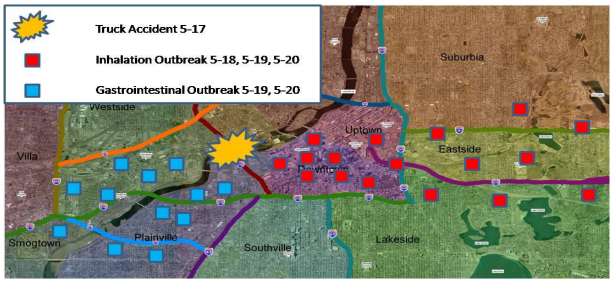


Figure 11: Ground Truth of Epidemic Spreads

Ground Truth: The ground truth provided by the Challenge committees is summarized in Figure 11. The ground zero location is a bridge over the river at the joint of midstream and downstream. On May 17, a truck carrying food contaminated by harmful spores had an accident there. The spores were dispersed into the air and the river, spread through wind and water current, and led to the outbreaks of inhalation and gastrointestinal diseases. Fever and diarrhea are two major symptoms, respectively. Due to the different incubation periods, the inhalation disease first outbreak on May 18 in Downtown, Uptown, and part of Plainville (revealed by patterns (5)–(8)). The disease also dispersed to the wide east on May 19–20 by west wind, but with much lower density. The gastrointestinal disease then outbreak on May 19 in downstream/southwest areas (revealed by patterns (12)–(15)). Unlike the *diarrhea* microblogs, the *fever* microblogs were collected in almost everywhere of the city, which makes the movement pattern noisy. Many people with the inhalation disease went to different hospitals on May 20 (revealed by patterns (9)–(11)). But people with the gastrointestinal disease never did, which is also verified by our results. In conclusion, CPM draws a complete and accurate picture of the epidemic outbreaks, and provides useful clues to infer the correct ground zero location.

4.3 Contextual Graph Mining

Data Overview: The dataset comes from the IEEE InfoVis 2007 Contest. It contains 20,204 US movies (2000–2006). For each movie, at least one director, one cinematographer and the first ten billed actors and actresses (at most 20 acting persons) are given with names. Movies are classified into 20 genres. Multiple genres are allowed. On average, one movie has 1.75 genres. IMDB and Netflix ratings, including the average rating score and the number of votes are also available. The movie collaboration graph is defined as such: a vertex represents a person, and an edge between two persons indicates they collaborate in at least one movie. We create two collaboration graph series: (1) by movie release year, and (2) by movie genre. Frequent subgraphs are mined on the two series, respectively. As graph mining cannot be done on the entire collaboration graph with up to 50,000 vertices and a million edges, we truncate the graphs by only considering the most popular (POP) and the best rated (BEST) movies, and also the persons that perform at least a certain number of movies. For example, in the Action movies, only the top 10 movies in the *number of user ratings* ranking (POP) and the top 10 movies in the *average rating score* ranking (BEST) are selected, respectively. The resulting four series of collaboration graphs are listed in Tables 9–10. Sample graph data is shown in Figure 12. Contextual information is defined over vertices (persons) by their acting preferences. For each person, all genres of the

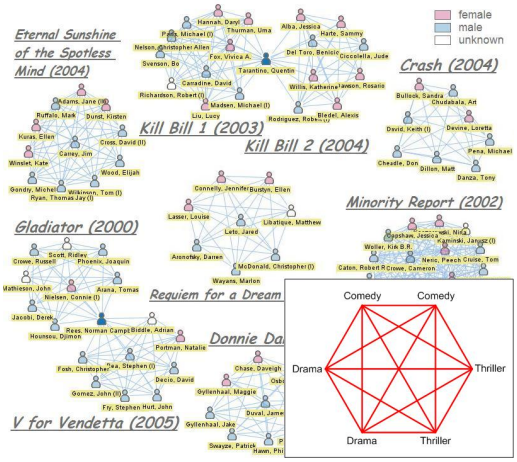


Figure 12: POP *Drama* Movies & CPM Result

Table 9: Movie Collaboration Graphs by Year

Year	Type	#vertex	#edge	Year	Type	#vertex	#edge
2000& 2001	POP	503	7299	2000& 2001	BEST	205	1790
2002& 2003	POP	546	10697	2002& 2003	BEST	357	5725
2004& 2005	POP	465	9509	2004	BEST	255	2108
				2005	BEST	428	8126

movies s/he has performed are collected. The top genres determined by either a large count or a large relative ratio are chosen as the person's acting genres. A CIG encoding the genre features of persons is shown in Figure 13.

Contextual Graph Mining: First we define the contextual subgraph isomorphism by conceptually extending the generalized subgraph isomorphism [9, 3]. We denote the vertex set of a graph g by $V(g)$, and the edge set by $E(g)$. A graph g is a contextual subgraph of another graph g' if there exits a contextual subgraph isomorphism from g to g'.

Definition 5 (Contextual Subgraph Isomorphism). *Given a CIG G, a contextual subgraph isomorphism is an injective function $\phi : V(g) \to V(g')$, s.t. (1) $\forall u \in V(g), \ell(u) = \ell'(\phi(u))$ or $\ell(u) \rhd \ell'(\phi(u))$ on G, and (2) $\forall (u,v) \in E(g)$, $(\phi(u), \phi(v)) \in E(g')$, $\ell(u,v) = \ell'(\phi(u), \phi(v))$ or $\ell(u,v) \rhd \ell'(\phi(u), \phi(v))$ on G, where ℓ and ℓ' are the label functions of g and g', respectively.*

The contextual graph mining is shown in Algorithm 4. Similar to the contextual association rule and sequence mining, here the over-generalized graph patterns [9, 3] need to be pruned: If a graph pattern gp is a contextual subgraph of a graph pattern gp' ($gp \neq gp'$) and their supports are identical, then gp is an over-generalized pattern.

Algorithm 4 Contextual Graph Mining

Input: Target graphs $\mathcal{D}$; CIG G; min_sup .
Output: Contextual graph pattern set $\mathcal{C}$.
1: find frequent contextual graphs with single vertex in $\mathcal{D}$;
2: expand the graphs by adding vertices or edges (labels can be contexts encoded in G) to generate new patterns;
3: insert those frequent (using min_sup) and non over-generalized patterns to $\mathcal{C}$;
4: **return** C;

Results: We apply a traditional graph mining algorithm, gSpan [20], a generalized graph mining algorithm, Taxogram [3], and CPM (Algorithm 4) to the movie graphs using min_sup = 100%: gSpan runs without any additional information, thus only finds subgraphs consisting of specific persons; Taxogram runs with a taxonomy that, for each person,

Table 10: Movie Collaboration Graphs by Genre

POP.Genre	#vertex	#edge	BEST.Genre	#vertex	#edge
Action	91	561	Action	70	432
Animation	63	223	Adventure	60	339
Comedy	77	396	Animation	35	115
Drama	94	482	Comedy	94	484
Horror	83	395	Crime	95	618
Musical	70	316	Documentary	60	416
Mystery	103	595	Drama	80	416
Romance	90	500	Fantasy	72	385
Sci-Fi	123	1240	Family	48	169
Thriller	101	632	Mystery	134	1078
War	97	607	Romance	84	461
			Sci-Fi	99	717
			Thriller	89	595
			War	80	485

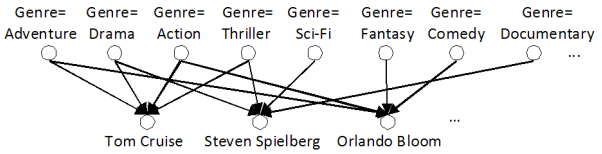


Figure 13: CIG Encoding Persons' Genre Contexts

a concept defined by his/her primary genre (with the largest relative ratio) is associated; CPM runs with the CIG in Figure 13. Results are summarized in Table 11. On the four graph series, gSpan only finds two subgraphs in total, suggesting that there is few consistent collaboration patterns at specific person level. By generalizing persons with the primary genre, Taxogram finds more patterns. With more contextual information available, CPM achieves the best results in name of the patterns discovered. It finds patterns more than 10 times of Taxogram in almost all graph series including several interesting ones: First, POP movies tends to have more common collaboration patterns than BEST movies in either the temporal investigation or the horizontal by-genre comparison. This suggests that there exists some common actor "flavor ingredient" to attract audience through advertising, across the years and the genres. In contrast, there are fewer common patterns for BEST movies. Every top-ranked movie may has its own reason for success. We further checked the discovered patterns among POP movies of different genres. In the 66 subgraphs with the largest size of 6, most are with the mixed person genre of Drama, Comedy and Thriller (Figure 12). This suggests that to produce a popular movie, there should at least have movie stars in these three genres to play to the gallery.

5. DISCUSSIONS

One may ask why simply joining the contextual information to the target dataset and then running a traditional single-source pattern mining algorithm does not work for CPM. The reason is two-fold: (1) The join can generate replicated contextual features for one target object, which makes the joined dataset contain redundancy and can result in meaningless patterns. For example, in Section 4.1, directly joining the *Road* information to the third row (tid=266) of Table 3 can generate two close_to(Road) predicates. If this occurs frequently in the crime target dataset, itemsets like {close_to(Road), close_to(Road)} can become frequent. But such patterns are meaningless. (2) The join will cause information loss that, the linkages between identities and the contextual information are removed. Again, take the case study in Section 4.1 as an example. Even if with removing the data redundancy caused by the join and Table 5 is obtained, itemsets like {within(Tract26), within(POPDEN=Low)} and {close_to(Trent Av), close_to(Road)} can be generated and frequent. However, such patterns reveals nothing new

Table 11: Collaboration Graph Mining Results

Graph Series	Type	Algorithm	#Patterns by Subgraph Size				
			2	3	4	5	6
By Year	POP	gSpan	2	0	0	0	0
		Taxogram	6	18	41	7	12
		CPM	49	497	3807	OoM*	OoM*
	BEST	gSpan	0	0	0	0	0
		Taxogram	2	7	35	272	1497
		CPM	27	187	966	OoM*	OoM*
By Genre	POP	gSpan	0	0	0	0	0
		Taxogram	3	4	4	1	0
		CPM	10	44	121	152	66
	BEST	gSpan	0	0	0	0	0
		Taxogram	3	8	6	0	0
		CPM	7	20	44	77	0

*Out of memory due to too many patterns discovered

Table 12: CPM vs. "Simple Join"

Dataset	Algorithm	CPU time (s)	#itemsets	#rules
crime	CPM	13.891	431	212
	Simple Join	14.016	822	1563

Dataset	Algorithm	CPU time (s)	Total #patterns	#patterns end with symptom
fever with min_sup =40%	CPM	152.672	659	9
	Simple Join	175.828	1334	12
diarrhea with min_sup =80%	CPM	14.266	537	15
	Simple Join	14.422	543	15
diarrhea with min_sup =40%	CPM	190.938	6877	68
	Simple Join	367.422	12932	76

but the known correlation between an item and its related feature, which is already represented in the CIG. It is a common issue among association rule, sequence, and graph mining. It is known that, if a small pattern, e.g., a meaningless 2-itemset in the above two cases, is frequent, it is bound to enlarge the search space by growing to larger patterns [16, 17, 9, 3], which inevitably leads to the generation of a huge number of redundant or over-generalized patterns. A feasible way to handle this is to post-process all the patterns after they are generated, by filtering the result with additional information. However, one may need to enumerate nearly all possible patterns due to its NP-completeness, which makes it intractable, especially when min_sup is small. Previous study [16] has shown that, not generating useless patterns in the earliest stage can make the algorithm 100 times faster than post-processing. On the other hand, the proposed CPM approach can avoid all these issues.

Table 12 shows the comparisons between CPM and running traditional pattern mining on the datasets simply joined with the contextual information (denoted by "Simple Join"). We can see that, with same parameters, "Simple Join" always costs more CPU time and finds a lot more patterns. However, none of the extra patterns is useful or provides new insight. The larger the dataset is and the smaller min_sup is, the more obvious that CPM is advantageous. In the case study of Section 4.3, graph mining without pruning over-generalized patterns yields little difference and thus is not shown here. Because the CIG is bipartite, a large number of over-generalized patterns can exist only when there are many specified patterns. However, from Table 11 we can see that, with min_sup = 100%, only 2 specified subgraph patterns, both of which contain 2 vertices, can be found (the first row, gSpan result). The largest possible number of over-generalized patterns of the two 2-vertex subgraphs is: (the number of contexts of one vertex) $\times$ (the number of contexts of the other) $-$ 1. In this specific case study, this number is small. Therefore, no significant difference can be observed between CPM and "Simple Join" here. However, we should keep in mind that, generally, pruning over-generalized contextual graph patterns is needed, especially when the number of specialized patterns is large.

There are two ways of handling missing values in CPM. Suppose feature F to be encoded has missing value. One way is to insert a vertex with label ($F = null$) to the CIG and CPM works. This may make ($F = null$) appear in the final patterns, which makes sense and is needed in some cases. The other way is to ignore all missing values when constructing the CIG, then they will not be considered in pattern mining. It is possible that the identities are not the same among different contextual information sources, which is not a problem for CIG encoding. As can be seen in Algorithm 1, CIG encoding can also be done incrementally. Whenever new contextual information sources are available, given a previously constructed CIG, new information can be encoded without re-encoding the previous information sources. The effects of different discretization methods to CIG encoding include the followings. Given a fixed size of identities $|V|$, a fine granularity of feature discretization can increase $|U|$ and $|E|$ of a CIG $G = (U, V, E)$, which can enlarge the search space for pattern mining because the number of frequent combinations can increase, and vice versa. Nonetheless, the linear time and space complexities of CIG encoding do not change at all.

6. RELATED WORK

Previous studies on frequent pattern mining mainly focus on mining from a single source. The most typical pattern mining methods include itemset/association rule mining [1, 8], sequential pattern mining [2, 19], and graph mining [20, 11]. In this paper, we propose the CPM approach that appropriately utilizes available contextual information from multiple sources for pattern discovery. The goals of CPM and multi-relational data mining (MRDM) approaches proposed to discover patterns involving multiple tables (relations) [4, 5] are similar, but they have different focuses: (1) CPM addresses to involve contextual information by non-intrusively utilizing existing single-source pattern mining methods, i.e., with a two-step "CIG encoding + pattern mining". The CIG encoding works with different types of pattern mining methods (itemset/associaiton rule, sequence, and graph), and can be seen as a unified pre-processing step, given that the contextual information sources link to the target dataset in a star schema. Because the encoding is efficient, with an efficient single-source pattern mining method implementation, the two-step strategy of CPM can be efficient and implemented with limited effort. This is especially meaningful for systems that already have a suite of single-source pattern mining algorithms seeking a quick and cheap enabling technique to collectively mine new information sources. On the other hand, implementing and deploying new algorithms is usually far more expensive in real-world systems. (2) Typical MRDM approaches solves the similar problem but in an intrusive way. To be specific, implementing MRDM approaches usually cannot reuse existing single-source pattern mining algorithms, especially when joining multiple tables is not considered a good solution. MRDM approaches based on inductive logic programming (ILP) [12, 4] can be applied to a general class of data mining problems and do not restrict to star schema of tables. Nonetheless, due to the generality, ILP based approaches can often be far less efficient in specific tasks compared with CPM. For example, WARMR [4] can be seen as a generalized algorithm for mining itemsets and sequences. Previous work [5] has pointed out that due to its general-purpose nature, WARMR does not fully exploit the properties of the specific patterns compared with traditional itemset and sequence

mining algorithms [1, 2] and their GPM versions [16, 17]. On the other hand, CPM based on the proposed CIG encoding framework works directly with the GPM algorithms designed for specific tasks, including association rule mining (Section 4.1), sequence mining (Section 4.2), and graph mining (Section 4.3). Therefore, we can infer that in most cases, the proposed CPM approach can have better efficiency than ILP based approaches when solving a same problem.

A different MRDM approach recently proposed is RMiner [15]. It is designed to mine the MCCS (Maximal Connected Complete Subgraph) pattern in the K-partite graph representation of a MRD (multi-relational database). First, multiple (say, K) tables in a MRD is converted into a K-partite graph. Then RMiner runs on the graph to find patterns in the form of MCCS. The differences between the proposed CPM approach and RMiner include: (1) CPM searches patterns of classical forms (itemset/association rule, sequence, and subgraph) in a space defined by the CIG and the target dataset (of transactions, sequences and graphs). RMiner searches MCCS patterns which is far different from classical forms, and on the K-partite graph defining its search space. (2) The proposed CIG encoding only encodes contextual information into a graph, which is always bipartite. The target dataset of transactions, sequences, or graphs is not changed a bit. In contrast, RMiner runs on the K-partite graph which actually encodes the entire MRD of all the K tables. Although both CPM and RMiner work with graph representations of information, they are significantly different. More fundamentally, RMiner has a different objective with traditional MRDM approaches that it finds interesting (defined by the maximum entropy model) pattern of co-occurring attributes [15], whereas CPM finds similar patterns to traditional MRDM approaches.

To obtain contextual patterns, other alternatives may also exist to the proposed CPM approach. However, they require either significant intrusive modifications to the pattern mining algorithms or the development of new algorithms, so that the contextual information can be handled appropriately. The CPM approach proposed in this paper, i.e., "CIG encoding + pattern mining", is the shortest path and the cheapest way so far we can find. As long as the applied GPM algorithms accept a general DAG structured CIG, no modification is required and CPM can be done transparently to the end users. This makes the system much easier to use, and little engineering effort is required.

7. CONCLUSIONS

With a growing number of information sources describing properties of related objects in today's big data challenge, we propose a general framework to encode contextual information from multiple sources via the Contextual Information Graph (CIG) for Contextual Pattern Mining (CPM). The main objective is the capability to reuse the existing single-source pattern methods without any modification and with high efficiency and ease. The proposed approach discovers more predictive, insightful, and interesting patterns that traditional pattern mining algorithms cannot find. The CIG encoding does not introduce any additional computational cost to the pattern mining process, except the pre-processing of the contextual data, which has linear time and space complexities. Missing values, multiple values, and dynamically changing contextual information can also be handled. We demonstrate through case studies that, the mining of all the three major types of contextual patterns, i.e., association rule, sequence, and graph, are well supported by our framework. Experiments and case studies on real-world datasets demonstrate that contextual patterns containing richer information and more useful insights are discovered by CPM, which is both efficient and easy to use; on the other hand, traditional methods cannot discover most of these interesting patterns.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, pages 487–499, 1994.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, pages 3–14, 1995.

[3] A. Cakmak and G. Ozsoyoglu. Taxonomy-superimposed graph mining. In *EDBT*, pages 217–228, 2008.

[4] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[5] S. Džeroski. Multi-relational data mining: an introduction. *SIGKDD Explorations*, 5(1):1–16, 2003.

[6] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *VLDB*, pages 420–431, 1995.

[7] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques, 3rd ed.* Morgan Kaufmann, 2006.

[8] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12, 2000.

[9] A. Inokuchi. Mining generalized substructures from a set of labeled graphs. In *ICDM*, pages 415–418, 2004.

[10] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Advances in spatial databases*, pages 47–66, 1995.

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, pages 177–187, 2005.

[12] S. Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[13] K. Olsen and A. Malizia. Automated personal assistants. *Computer*, pages 112–111, 2011.

[14] D. Page and M. Craven. Biological applications of multi-relational data mining. *SIGKDD Explorations*, 5(1):69–79, 2003.

[15] E. Spyropoulou and T. De Bie. Interesting multi-relational patterns. In *ICDM*, pages 675–684, 2011.

[16] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995.

[17] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, pages 1–17. Springer, 1996.

[18] H. Wang, H. He, J. Yang, P. Yu, and J. Yu. Dual labeling: Answering graph reachability queries in constant time. In *ICDE*, pages 1–12, 2006.

[19] J. Wang, J. Han, and C. Li. Frequent closed sequence mining without candidate maintenance. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1042–1056, 2007.

[20] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM*, pages 721–724, 2002.