Mining Interesting Patterns in Multi-relational Data with N-ary Relationships

Eirini Spyropoulou[1], Tijl De Bie[1], and Mario Boley[2]

[1] Intelligent Systems Lab, University of Bristol, UK
[2] Fraunhofer, IAIS, Germany

Abstract. We present a novel method for mining local patterns from multi-relational data in which relationships can be of any arity. More specifically, we define a new pattern syntax for such data, develop an efficient algorithm for mining it, and define a suitable interestingness measure that is able to take into account prior information of the data miner. Our approach is a strict generalisation of prior work on multi-relational data in which relationships were restricted to be binary, as well as of prior work on local pattern mining from a single n-ary relationship. Remarkably, despite being more general our algorithm is comparably fast or faster than the state-of-the-art in these less general problem settings.

1 Introduction

Pattern mining research has mostly focused on mining itemsets or association rules on one binary table representing transactions and items. Real world data is often more complex, commonly stored in a relational database that allows to capture the relations that exist among the different entity types in the data.

Mining multi-relational data has for a long time been approached by Inductive Logic Programming (ILP) methods where the goal is to mine frequent rules about the data [5,12]. More recently, there have also been generalisations of itemset mining to more complex settings.

Itemset mining has been generalised to mining local patterns in a (single) relationship that involves several entity types. Such a relationship is known as *n-ary relationship* [3,8,9]. An example of a 3-ary relationship is a shop transaction record containing the item sold, the customer, as well as the mode of payment used (e.g. cash/debit card/credit card). The definition of a closed itemset can be generalised fairly directly to n-ary relationships. Algorithms designed to mine such patterns typically require choosing a frequency threshold for each entity type, and do not define an interestingness to rank the patterns found.

Another strand of research has generalised itemset mining to *multi-relational data*. Think for example of a retail database which contains customers, items they bought as well as characteristics of customers and characteristics of items. The most common strategy has been to essentially apply frequent itemset mining on the join of all tables of the database [11,7,10]. Although this strategy seems sensible, in previous work [13,14] we argued that it has significant disadvantages.

J. Fürnkranz, E. Hüllermeier, and T. Higuchi (Eds.): DS 2013, LNAI 8140, pp. 217–232, 2013.

We thus proposed the so-called Maximal Complete Connected Subset (MCCS) pattern syntax as an alternative, which is a natural generalisation of closed itemsets and their supporting transactions. We also proposed a global measure of subjective interestingness. While we were able to demonstrate promising results, a restriction is that the method can be applied to multi-relational data only if all relationships in the database are binary.

Contributions in this paper. Although both research strands have remained largely independent, real data is often multi-relational while also involving relationships of arity larger than 2. To the best of our knowledge no local pattern mining method exists targeting this case, and we aim to fill this gap in the current paper. In doing this, we wanted to ensure backward compatibility of the newly introduced notions with both these strands of research, ideally without sacrificing computational tractability. More specifically, we introduce a novel pattern syntax called N-MCCSs and show that it is a generalisation of pattern syntaxes defined for simpler settings; namely of MCCSs for multi-relational data with binary relationships [13,14], as well as of n-Sets for data represented by a single n-ary relationship [3].

Developing an algorithm to mine N-MCCSs proved non-trivial owing to the differences in algorithmic approaches for both strands of research. Nonetheless, our experiments show that the proposed algorithm performs similarly to the algorithm for mining MCCSs from multi-relational data with binary relationships only [14], and on n-ary data it considerably outperforms Data-Peeler [3], the state-of-the-art algorithm for mining data containing a single n-ary relationship. Thus, our contributions simultaneously *generalize and unify important previous data mining methods* and *approximately match or even improve on them in terms of computation times*.

2 Definitions and Concepts

Here we formalize multi-relational data as considered in this paper and define the concepts needed to introduce the pattern syntax we propose.

2.1 Multi-relational Data

Our formalization of multi-relational data follows the Entity-Relationship (ER) model [6], with the difference that we treat every attribute as an entity type of its own. This unified treatment of entity types and attributes leads to a simple formalization which still captures all the associations in the data.

We formalize **multi-relational data** as $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$. E is a finite set of **entities** which is partitioned into k **entity types**. We formalize this by defining a mapping from the entity set onto an index set for the types: $\mathbf{t} : E \to T = \{1 \ldots, k\}$. Thus, the set of entities is the union of the sets of entities for each type: $E = E_1 \,\dot\cup\, \ldots \,\dot\cup\, E_k$ with $E_i = \{e \in E \mid \mathbf{t}(e) = i\}$. Slightly abusively, we will also apply $\mathbf{t}$ to sets of entities, to yield a set with their entity types.

We also define a **relationship type** R as a set of entity types, such that the set of all relationship types $\mathbf{R}$ is a subset of the power set of T, i.e.: $\mathbf{R} \subseteq \mathcal{P}(T)$. For each $R \in \mathbf{R}$ there is an $|R|$-ary relationship $\mathcal{R}_R \subseteq \{\{e_1, \ldots, e_n\} : R = \{\mathbf{t}(e_1), \ldots, \mathbf{t}(e_n)\}\}$, containing all sets of entities of the types in R that are indeed related with each other. We say that a set $\{e_1, \ldots, e_n\} \in \mathcal{R}_R$ is a **relationship instance**. The set $\mathcal{R}$ is the union of all relationships, i.e., $\mathcal{R} = \bigcup_{R \in \mathbf{R}} \mathcal{R}_R$. Figure 1 shows a toy example of some multi-relational data highlighting the notions we just introduced.

Bibliographic multi-relational data

tagged_by_user

user	tag	paper
U1	algorithms	P1
U1	large graphs	P2
U1	algorithms	P2
U2	large graphs	P1
U2	large graphs	P2

authored_by

paper	author
P1	Jure Lescovec
P1	Christos Faloutsos
P2	Jure Lescovec
P2	Christos Faloutsos

Fig. 1. Bibliographic multi-relational data containing four entity types ("user", "tag", "paper", "author") and two relationship types. One 3-ary between "user", "tag" and "paper" and one binary between "paper" and "author".

2.2 The Pattern Syntax of N-MCCSs

The pattern syntax proposed in this paper is called N-MCCS and builds upon two previously considered ones: the pattern syntax of MCCSs, which was proposed for the case of multi-relational data with many binary relationships and is a generalisation of closed itemsets or maximal tiles [13,14], and the pattern syntax of n-Sets, which was proposed for the case of data with one n-ary relationship only [3]. In this section we define N-MCCSs and in the next section we show that it is indeed a generalisation of MCCSs and n-Sets.

Intuitively, N-MCCSs capture associations between entities of different entity types. In the example of Fig. 1, an N-MCCS could correspond to a group of users that have tagged the same group of papers of the same authors or a group of users that have tagged the same group of papers with the same tags.

The definition of N-MCCSs is based on the notions of *connectedness* and *completeness*. A set of entities is connected if any pair of entities in it is connected through a sequence of entities that are pairwise related. Formally:

Definition 1. *We call a set of entities $F \subseteq E$* ***connected*** *if for all distinct $e, e' \in F$ there is a sequence $e = e_1 \ldots e_l = e'$ with $\{e_1..e_l\} \subseteq F$ such that for $i \in \{1, \ldots, l-1\}$ it holds that there is an $F' \supseteq \{e_i, e_{i+1}\}$ such that $F' \in \mathcal{R}$.*

Example 1. In the toy data example of Fig. 1 the set of entities {U1, U2, large graphs, P2, Christos Faloutsos} is connected whereas the set of entities {U1, U2, large graphs, Christos Faloutsos} is not connected because for none of the pairs {U1, Christos Faloutsos}, {U2, Christos Faloutsos}, {large graphs, Christos Faloutsos} is there a superset which is a subset of any relationship.

A set of entities is complete if every subset containing entities of different entity types that belong to the same relationship type, is a subset of a relationship instance. One might expect that for completeness we would require all entity types of a relationship type to be present in a pattern. However our definition of completeness is more flexible. In the example of Fig. 1 for instance it allows patterns containing only a set of papers and their tags without containing the users. We formally define completeness using the notion of a critical subset.

Definition 2. *Given some relational data* $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$*, and a set of entities* $F \subseteq E$*, a set* $F' \subseteq F$*, with* $|\boldsymbol{t}(F')| = |F'|$ *and* $\boldsymbol{t}(F') \subseteq R \in \boldsymbol{R}$ *is called a* ***critical subset*** *of* F *with respect to* R*. We say that a critical subset,* F' *of* F*,* ***is covered*** *in* $\mathcal{R}$ *if for all* $R \in \boldsymbol{R}$ *with respect to which it is critical, there exists an* $F'' \supseteq F'$ *such that* $F'' \in \mathcal{R}_R$*.*

Definition 3. *Given some relational data* $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$*, we call a set of entities* $F \subseteq E$ ***complete*** *iff all critical subsets of* F *are covered in* $\mathcal{R}$*.*

Example 2. Considering again the data toy example of Fig. 1 the set{U1, U2, large graphs, Christos Faloutsos} is complete but not connected.

A set of entities $F \subseteq E$ is called a Complete Connected Subset and abbreviated as **N-CCS** if it satisfies both completeness and connectedness as defined here for multi-relational data of n-ary relationships. Like in other pattern mining tasks the set of N-CCSs is typically exponentially larger than the input size. For reasons of efficiency of the mining algorithm, we focus on a smaller subset of the N-CCSs namely the maximal N-CCSs. A maximal N-CCS, abbreviated as **N-MCCS**, is an N-CCS to which no entity can be added without violating connectedness or completeness.

Example 3. Two examples of N-MCCSs from the toy data of Fig. 1 are: {U1, U2, large graphs, P2, Christos Faloutsos, Jure Lescovec} and {P1, P2, large graphs, algorithms, Christos Faloutsos, Jure Lescovec}.

2.3 From MCCSs and n-Sets to N-MCCSs

Here we show that the proposed pattern syntax of N-MCCSs is a generalisation of MCCSs [13,14], as well as n-Sets [3]. We do this by translating the definition of MCCSs and n-Sets using the concepts of this paper.

MCCSs are Maximal Complete Connected Subsets where completeness and connectedness are defined as follows. For $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ where $\forall R \in \mathbf{R}$, $|R| = 2$, a set $F \subseteq E$ is complete if $\forall F' \subseteq F$ with $t(F') \in \mathbf{R}$ and $|F'| = 2$ it holds that $F' \in \mathcal{R}_{t(F')}$. A set $F \subseteq E$ is connected if for all $e, e' \in F$ there is a sequence $e = e_1, \ldots, e_l = e'$ with $\{e_1, \ldots, e_l\} \subseteq F$ such that for $i \in \{1, \ldots, l-1\}$ it holds that $\{e_i, e_{i+1}\} \in \mathcal{R}$. Clearly both completeness and connectedness are special cases of completeness and connectedness as defined in Def. 3 and Def. 2 respectively.

n-Sets are defined as follows. For some multi-relational data $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ where $|\mathbf{R}| = 1$ and for $R \in \mathbf{R}$, $|R| = n$, a set $F \subseteq E$ with $|\mathbf{t}(F)| = n$ is an

n-Set if $\forall F' \subseteq F$ with $|\mathbf{t}(F')| = |F'| = n$, $F' \in \mathcal{R}_R$ and it is maximal under this condition. Since $|\mathbf{R}| = 1$, and for every n-Set F, $|\mathbf{t}(F)| = n$, if F is complete it will be connected as well. Therefore the pattern syntax of N-MCCSs is a generalisation of the pattern syntax of n-Sets.

3 Mining Algorithm

In Sec. 2.2 we argued that for efficiency reasons we focus on mining maximal patterns. An efficient algorithm to mine all N-MCCSs should avoid enumerating most N-CCSs that are not maximal. The algorithm proposed here, is an instantiation of the fixpoint listing algorithmic framework [2] which is used to enumerate all N-CCSs that are fixpoints of a closure operator and to which we refer as *closed N-CCSs*. In Sec. 3.1 we establish the applicability of this framework for mining N-MCCSs. Then, in Sec. 3.2 we describe the fixpoint listing framework and in Sec. 3.3 we instantiate it for our setting by defining a suitable closure operator. In Sec. 3.4 we also show that this operator is a generalisation of the closure operator which was defined within the same algorithmic framework for the case of MCCSs [14] and in Sec. 3.5 we discuss implementation details.

3.1 The Applicability of the Fixpoint Listing Framework

The divide-and-conquer fixpoint listing algorithm enumerates all closed sets of a closure operator from a given set system. A **set system** is a family of subsets $\mathcal{F} \subseteq \mathcal{P}(A)$ over some ground set A, where $\mathcal{P}(A)$ the power set of A. A set $F \in \mathcal{F}$ is called *closed* if it is a fixpoint of some **closure operator** $\rho : \mathcal{F} \to \mathcal{F}$, i.e., $\rho(F) = F$. For the operator ρ to be a closure operator it must satisfy three properties: extensivity ($F \subseteq \rho(F)$ for all $F \in \mathcal{F}$); monotonicity ($\rho(F) \subseteq \rho(F')$ for all $F, F' \in \mathcal{F}$ with $F \subseteq F'$); and idempotence ($\rho(\rho(F)) = \rho(F)$ for all $F \in \mathcal{F}$).

In order for the fixpoint algorithmic framework to be applicable for mining N-MCCSs, the set of closed N-CCSs needs to be a superset of the set of N-MCCSs. This is guaranteed by the properties of the closure operator. If an N-CCSs F is maximal it means that it cannot be extended by any other entity. From the extensivity of the closure operator and from the fact that $\rho(F) \in \mathcal{F}$, if follows that $\rho(F) = F$. Which means that F is a closed N-CCS.

In order for the framework to be applicable for mining the closed sets of a set system, this set system must satisfy a particular property called *strong accessibility* [2]. We now prove that independent of the input data $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ the corresponding **set system of N-CCSs**, defined as

$$\mathcal{F}_{\mathbb{D}} = \{F \subseteq E \mid F \text{ connected } \wedge F \text{ complete}\}$$

is always strongly accessible. For a set system $\mathcal{F} \subseteq \mathcal{P}(A)$ and a set $F \in \mathcal{F}$, let us denote by $Aug(F) = \{a \in A \mid F \cup \{a\} \in \mathcal{F}\}$ the set of valid **augmentation elements** of F. Then $\mathcal{F}$ is called **strongly accessible** if for all $X \subset Y \subseteq A$ with $X, Y \in \mathcal{F}$ there is an element $e \in (Aug(X) \setminus X) \cap Y$.

Theorem 1. *For all relational data* $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$*, the set system* $\mathcal{F}_{\mathbb{D}}$ *of N-CCSs is strongly accessible.*

In [14] we showed that for multi-relational data containing just binary relationships the set system of Complete Connected Subsets (CCSs) is strongly accessible. The same argumentation line can be used to prove Theorem 1 as well.

3.2 N-RMiner

As we have already established that the divide and conquer fixpoint listing algorithmic framework of [2] is applicable to the case of N-CCSs, we now describe the instantiation of the framework for this setting.

Algorithm 1. N-RMiner: List all N-MCCSs

N-RMiner(F, B)
1: Select $e \in Aug(F) \setminus (F \cup B)$
2: $F' = g(F \cup \{e\})$
3: **if** $F' \cap B = \emptyset$ **then**
4: **if** $F' = Aug(F')$ **then**
5: Output F'
6: **else**
7: **N-RMiner**(F', B)
8: **end if**
9: **end if**
10: **N-RMiner**$(F, B \cup \{e\})$

The general structure of this divide-and-conquer algorithm is shown in Algorithm 1. In each recursive call, the algorithm selects an element e from the set of valid augmentation elements, $Aug(F)$, of the current solution F(line 1), and splits the search space into two subtrees: one subtree in which all N-CCSs include the element e (lines 3-9) and another subtree in which all N-CCSs exclude e (line 10). This is achieved by adding it to B, which represents the set of elements already considered as extensions to F (line 10). The fact that only closed patterns are sought is ensured in line 2, where the expanded set $F \cup \{e\}$ is potentially further expanded by applying the operator g (see bellow for a definition).

As N-RMiner enumerates all closed N-CCSs, we added lines 4-5 to ensure it outputs N-MCCSs only, i.e., N-CCSs F for which there are no augmentation elements not yet in F. Formally this is verified by checking if $F = Aug(F)$.

As defined in Sec. 3.1, the set of augmentation elements $Aug(F)$ of a set $F \in \mathcal{F}_{\mathbb{D}}$ from a set system is the set of all elements that can be individually added to F to yield another set from the same set system. Specifically for the set system $\mathcal{F}_{\mathbb{D}}$ of N-CCSs, and given a relational database $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$, the set $Aug(F)$ corresponds to the following set: $Aug(F) = \{e \in E \mid F \cup \{e\}$ is complete and connected$\}$.

In [14] we presented a useful additional feature of the algorithm which gives the option of focusing on a smaller set of patterns by defining additional constraints on the pattern syntax. We defined constraints on the minimum number of entities per entity type (minimum coverage constraint) as well as an upper bound which can safely be used for pruning as the set system still remains strongly accessible. The same type of constraints can be used here exactly in the same way.

3.3 The Closure Operator

Since Algorithm 1 is based on adding elements from $Aug(F)$ one by one, it is important to introduce the notion of compatibility of an element with a set, upon which the definition of the closure operator g is based.

Definition 4. *For some relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$, we say that an entity e is **compatible** with a set $G \subseteq E$ and denote $e \parallel G$, iff all critical sets $F' \subseteq (G \cup \{e\})$ with $e \in F'$, are covered in $\mathcal{R}$.*

We call the set of all $e \in E$ that are compatible with a set $F \in \mathcal{F}_{\mathbb{D}}$ the **set of compatible entities** of F and denote it as $Comp(F)$. We note here that the compatibility property is anti-monotone. This means that for $F, F' \subseteq E$ with $F \subseteq F'$, if for $e \in E$ it holds that $e \parallel F'$ then $e \parallel F$. Therefore also for $F, F' \in \mathcal{F}_{\mathbb{D}}$ with $F \subseteq F'$, $Comp(F') \subseteq Comp(F)$.

We would like to note here that for entities of type $i \notin \mathbf{t}(Aug(F))$ there are no critical sets between them and F and therefore by definition of compatibility $Comp(F) \cap E_i = E_i$.

Lemma 1. *Let $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ some relational data. A set $F \subseteq E$ is complete if and only if for all $e \in F$ it holds that $e \parallel F$.*

Proof. If an $F \subseteq E$ is complete then all the critical subsets of F are covered in $\mathcal{R}$. Since $e \in F$, we have that all critical subsets of $F \cup \{e\}$ are covered, and therefore $e \parallel F$. Conversely, for an $F \subseteq E$ such that for all $e \in F$ it holds that $e \parallel F$, we have that for all $e \in F$ all critical subsets containing e are covered in $\mathcal{R}$, which means that all critical subsets of F are covered. Thus F is complete.

It follows from Lemma 1 that for a set $F \in \mathcal{F}_{\mathbb{D}}$, $F \subseteq Comp(F)$. It also follows that $Aug(F) \subseteq Comp(F)$, because for every element $e \in Aug(F)$, $e \parallel F$ always holds. Therefore $F \subseteq Aug(F) \subseteq Comp(F)$.

From Lemma 1 it also follows that for $F \in \mathcal{F}_{\mathbb{D}}$, $Comp(F)$ can equivalently be defined as the set of all $e \in E$ such that $F \cup \{e\}$ is complete.

We are now able to define the operator g. The definition of g requires that every element in the closure of F is compatible with all the compatible elements of F. Formally:

$$g(F) = F \cup \{e \in Aug(F) \setminus F : e \parallel Comp(F)\}, F \in \mathcal{F}_{\mathbb{D}}.$$

Before we show that g is a closure operator by proving each of the essential properties, we give an example of applying g to a set.

Example 4. Let us consider again the toy dataset of Fig. 1. Let us also consider that $F = \{$P2, Christos Faloutsos$\}$. Then $Aug(F) \setminus F = \{$P1, U1, U2, large graphs, algorithms, Jure Lescovec$\}$ and $Comp(F) = \{$P1, P2, U1, U2, large graphs, algorithms, Jure Lescovec, Christos Faloutsos$\}$ and $g(F) = \{$P2, Christos Faloutsos, Jure Lescovec$\}$. We see that from all the elements of $Aug(F) \setminus F$ only "Jure Lescovec" is compatible with $Comp(F)$ because all critical sets, namely {P1,Jure Lescovec} and {P2,Jure Lescovec}, are covered.

Proposition 1. *For all $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ the codomain of g is the set system $\mathcal{F}_{\mathbb{D}}$ of N-CCSs, i.e., for all $F \in \mathcal{F}_{\mathbb{D}}$ it holds that $g(F) \in \mathcal{F}_{\mathbb{D}}$.*

Proof. We need to show that for every $F \in \mathcal{F}_{\mathbb{D}}$, $g(F)$ is complete and connected. When $F = \emptyset$ then $g(F) = \emptyset$ and therefore it is complete and connected. We show this is the case as well for all $F \neq \emptyset$, $F \in \mathcal{F}_{\mathbb{D}}$.

We have that $g(F) \subseteq Aug(F)$. From the definition of the set $Aug(F)$ this means that for every $e \in g(F)$, $F \cup e$ is connected. This means that there is a sequence of pairwise related entities between every pair $e, f \in g(F)$. Therefore $g(F)$ is connected.

To show completeness let us assume that $g(F)$ is not complete. This means that there is a critical subset $F' \subseteq g(F)$, that is not covered. We have that F is complete since $F \in \mathcal{F}_{\mathbb{D}}$. This means that $F' \not\subseteq F$ and therefore there must be at least one $e' \in F'$ such that $e' \in \{e \in Aug(F) \setminus F : e \parallel Comp(F)\}$. We therefore have that all critical subsets of $Comp(F)$ that contain e' are covered. Since $F' \subseteq g(F) \subseteq Comp(F)$, the fact that F' is not covered is a contradiction. Therefore $g(F)$ is complete.

It is trivial to see that g is extensive, as it does not remove any elements from the set it is applied to. Next we will prove that g is also monotone and idempotent (as long as the data satisfies some properties).

Proposition 2. *For all $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ the operator g is monotone.*

Proof. Assume the operator is not monotone, i.e., there are $F', F \in \mathcal{F}_{\mathbb{D}}$ with $F \subseteq F'$ such that $g(F) \not\subseteq g(F')$. Hence, there is an $e \in g(F) \setminus g(F')$. We also have $e \in g(F) \setminus F$ because g is extensive. From the definition of g it follows $e \parallel Comp(F)$. By anti-monotonicity of compatibility, $F \subseteq F'$ implies that $Comp(F) \supseteq Comp(F')$. Applying once again the monotonicity of compatibility, $e \parallel Comp(F)$ also implies $e \parallel Comp(F')$. Since $F' \subseteq Comp(F')$ we have that $e \parallel F'$. Also $e \in (Aug(F) \setminus F)$ and since F' is a connected superset of F, e is connected to F', and thus $e \in Aug(F')$. Hence $e \notin g(F')$ is a contradiction.

Before we give the proposition about idempotence, we define a class of multi-relational data called **acyclic multi-relational data** as all $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ such that there does not exist a sequence of entity types $\{i_1 \dots i_l\}$ such that for $j = 1 \dots l-1$, $\{i_j, i_{j+1}\} \subseteq R \in \mathbf{R}$ and $i_1 = i_l$.

Proposition 3. *For all acyclic $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ with the property that $\nexists e \in E$ such that $\exists R \in \boldsymbol{R}$ with $e \parallel (\cup_{i \in R} E_i)$, the operator g is idempotent.*

Proof. Assume that for an acyclic $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$, such that $\nexists e \in E$ such that $\exists R \in \mathbf{R}$ with $e \parallel (\cup_{i \in R} E_i)$, the operator g is not idempotent. This means that $\exists f \in g(g(F))$ such that $f \notin g(F)$. This can happen if:

(a) $f \nparallel Comp(F)$. Since $f \in g(g(F))$, $f \parallel Comp(g(F))$. For all $e \in g(F)$ we have that $e \parallel Comp(F)$. This means that e is compatible with all the compatible elements of F including F itself. Therefore adding e to F does not change this set which means that for every $e \in g(F)$, $Comp(F \cup \{e\}) = Comp(F)$. Therefore $Comp(g(F)) = Comp(F)$ and as a result $f \parallel Comp(F)$. A contradiction.

(b) $f \notin Aug(F)$. From the anti-monotonicity of compatibility and $f \parallel Comp(F)$, it holds that $f \parallel F$. Therefore for $f \notin Aug(F)$ it must be that f is not connected to F. Therefore $\mathbf{t}(f) \notin \mathbf{t}(Aug(F))$ which means that $\mathbf{t}(f)$ belongs to a relationship type R of which no entity types are in F. However, $\mathbf{t}(f) \in \mathbf{t}(Aug(g(F)))$. Therefore there must be an $e \in g(F)$ such that there is a set F', with $\mathbf{t}(F') \subseteq R \in \mathbf{R}$, $e, f \in F'$ and $F' \in \mathcal{R}$. Since $e \in g(F)$ we have that $e \parallel Comp(F)$. Since $\mathbb{D}$ is acyclic, no entity type of R except $\mathbf{t}(e)$ belongs to $Aug(F)$. Therefore we have that for every $i \in R$, $i \neq \mathbf{t}(e)$, $Comp(F) \cap E_i = E_i$. Therefore, for $e \parallel Comp(F)$ it must be that $e \parallel (\cup_{i \in R} E_i)$ (since e is compatible to the entities of the same type as well). A contradiction.

From the above propositions we have the following corollary:

Corollary 1. *For all acyclic multi-relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ with the property that $\nexists e \in E$ such that $\exists R \in \boldsymbol{R}$ with $e \parallel (\cup_{i \in R} E_i)$, the operator g is a closure operator.*

We note here that even though the idempotency of the operator g holds for acyclic multi-relational data that do not contain any entity belonging to all the relationship instances of a relationship, our method can still be applied to any multi-relational data by introducing an operator applying g to a set F as many times as required until the mapped set does not increase any more.

3.4 The Generality of the Closure Operator

The fixpoint listing algorithmic framework has also been used for mining MCCSs in multi-relational data with binary relationships. The instantiated algorithm for this setting is called RMiner [14]. Here we are going to show that the closure operator g is more general than the closure g_2 defined in R-Miner as:

$$g_2(F) = \{e \in Aug(F) : Comp'(F \cup \{e\}) = Comp'(F)\}$$

The set of compatible elements for a set $F \in \mathcal{F}_{\mathbb{D}}$ is defined in [14] as $Comp'(F) = \{e \in E \mid F \cup \{e\} \text{ is complete}\}$. In the conclusions of Lemma 1 we said that this is an equivalent definition for the set $Comp$ that we defined in this paper. Therefore $Comp$ and $Comp'$ map to the same set.

Following from that the pattern syntax of N-MCCSs is a generalisation of the pattern syntax of MCCSs it also holds that the codomain of the two closure operators is the same. We can now state the following theorem.

Theorem 2. *For all relational data* $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$*, such that* $\forall R \in \boldsymbol{R}$*,* $|R| = 2$*, the closure operator* $g : \mathcal{F}_{\mathbb{D}} \to \mathcal{F}_{\mathbb{D}}$ *is a generalisation of the closure operator* $g_2 : \mathcal{F}_{\mathbb{D}} \to \mathcal{F}_{\mathbb{D}}$ *[14], i.e.,* $\forall F \in \mathcal{F}_{\mathbb{D}}$*,* $g(F) = g_2(F)$*.*

Proof. We are first going to prove that $g(F) \supseteq g_2(F)$. Let us assume the opposite, i.e., $\exists e \in g_2(F)$ such that $e \notin g(F)$. $e \notin g(F)$ means that $e \notin F$ and $e \nparallel Comp(F)$. Therefore there is at least one $e' \in Comp(F)$ such that $\{e, e'\} \notin \mathcal{R}$ which means that $Comp(F \cup \{e\}) \subset Comp(F)$. Contradiction.

Now we are going to prove that $g(F) \subseteq g_2(F)$. Let us assume the opposite, i.e., that $\exists e \in g(F)$ such that $e \notin g_2(F)$. $e \notin g_2(F)$ means that $Comp(F \cup \{e\}) \subset Comp(F)$ and therefore there exists at least one $e' \in Comp(F)$ such that $\{e, e'\} \notin \mathcal{R}$. This means that $e \nparallel Comp(F)$ which is a contradiction since we have assumed that $e \in g(F)$. Therefore $g(F) = g_2(F)$.

The result presented in Theorem 2 is very important as it means that when N-RMiner is applied to data with binary relationships only, it enumerates exactly the same set of patterns as RMiner which is using the operator g_2.

3.5 Implementation Details

Our implementation is based on giving an id to every relationship instance and storing the relationship instance ids, in a global structure containing for every entity and every relationship type, a list of relationship instance ids. Then the coverage of a critical subset can easily be checked by checking whether the intersection of the relationship instance ids of the entities it contains is non-empty. Although to check the compatibility of an element with a set all critical sets need to be checked for coverage, it can be done more efficiently as it suffices for one critical set not to be covered in order for the compatibility not to hold.

The choice of the next element to extend the current partial solution is implemented with a for loop using an ordering of the entities in terms of increasing $Aug(e)$. This way the number of closure checks is reduced.

4 Assessment of Patterns

We assess the quality of a pattern based on a subjective interestingness measure which formalises how interesting a pattern is based on the prior information of the user. We adopt a definition of interestingness which was proposed for binary tables and adapted for binary related multi-relational data as well in our previous work [4,13]. According to this definition interestingness is defined as unexpectedness with respect to a Maximum Entropy model of the prior information of the user. In what follows we describe the type of prior information we consider in this paper and formalize and derive the Maximum Entropy optimization problem. We then give the definition of the interestingness measure.

4.1 The Maximum Entropy Model

In this paper we consider as prior information the number of relationship instances an entity participates in. This intuitively means that a pattern containing entities that appear less frequently in relationship instances will be deemed more interesting by our measure. As an example consider a movie database consisting of three entity types, "person", "role" and "movie" and a 3-ary relationship between them. According to this prior information, a pattern containing persons that appear in many films under any role will be rendered less interesting than a pattern containing people that appear less often as the later is more unexpected.

We formalize this prior information by fitting the Maximum Entropy (MaxEnt) distribution P on the data, with constraints on the expected sum of the relationship instances every entity participates in, being equal to the actual sum of relationship instances it participates in. Without details, we point out that a similar reasoning as in [13,14] establishes that this MaxEnt model P is a product distribution, with a factor P_R for each relationship type—so here we focus on how to obtain a MaxEnt model for just one n-ary relationship type.

For convenience, let us represent each n-ary relationship $\mathcal{R}_R$ (with $|R| = n$) of some multi-relational data $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ by means of an n-dimensional binary valued tensor $\mathbf{D}_R$. The dimensions of $\mathbf{D}_R$ are indexed by the entity types $t \in R$, and each dimension itself is indexed by an index i_t running over all entities of type $t \in R$. We denote as e_{i_t} the i'th entity of type t. The relationship $\mathcal{R}_R$ is encoded in $\mathbf{D}_R$ by ensuring that $\mathbf{D}_R(i_1, \ldots, i_n) = 1$ iff $\{e_{i_1}, \ldots, e_{i_n}\} \in \mathcal{R}_R$. Then the prior information on the relationship type R can be expressed as:

$$\sum_{\mathbf{D}_R} P_R(\mathbf{D}_R) \sum_{(i_1 \ldots i_n): i_t = k} \mathbf{D}_R(i_1, \ldots, i_t, \ldots, i_n) = d_R^{t,k}, (\forall t \in R)$$

where $d_R^{t,k}$ is the sum of all elements in $\mathbf{D}_R$ for which $i_t = k$ or in other words the number of relationship instances that the kth entity participates in. Finally, being a probability distribution, P_R needs to be normalised: $\sum_{\mathbf{D}_R} P_R(\mathbf{D}_R) = 1$.

The MaxEnt optimization problem maximizes $\sum_{\mathbf{D}_R} -P_R(\mathbf{D}_R) \log(P_R(\mathbf{D}_R))$, the entropy, with respect to P_R, subject to the above constraints. Computing the Karush Kuhn Tucker optimality conditions, similar to [4], shows that the optimal solution is of the form:

$$P_R(\mathbf{D}_R) = \prod_{i_1 \ldots, i_n} P_R^{\{i_1, \ldots, i_n\}}(\mathbf{D}_R(i_1, \ldots, i_n)),$$

$$\text{where } P_R^{\{i_1, \ldots, i_n\}}(d) = \frac{\exp\left(d \cdot \sum_t \lambda_R^{t,i_t}\right)}{1 + \exp\left(\sum_t \lambda_R^{t,i_t}\right)}.$$

Here, λ_R^{t,i_t} is a Lagrange multiplier corresponding to the constraint for entity i_t of type t. Thus, every factor P_R of the distribution P, is itself a product of independent Bernoulli distributions for the elements of the tensor. The optimal value of the Lagrange multipliers can be found by solving the convex Lagrange dual optimization problem ([4] gives details for the binary case).

4.2 The Interestingness of a Pattern

The proposed interestingness measure is a trade-off between the self-information of a pattern, which measures how much information a pattern carries, and the description length of a pattern, which measures how concisely it conveys this information. Intuitively an end user wants to see patterns that convey as much information as possible, as concisely as possible. Thus, interestingness of an N-MCCS $F \in \mathcal{F}_{\mathbb{D}}$ is defined as follows:

$$\text{Interestingness}(F) = \frac{\text{SelfInformation}(F)}{\text{DescriptionLength}(F)}.$$

Let us define the set of maximal critical subsets of F, denoted as $\mathcal{M}(F)$, as the set of all $F' \subseteq F$ such that $|\mathbf{t}(F')| = |F'|$ and $\mathbf{t}(F') \subseteq R \in \mathbf{R}$ and $\nexists F'' \subseteq F$ with $\mathbf{t}(F') \subset \mathbf{t}(F'') \subseteq R$. The self-information of a pattern F is given by:

$$\text{SelfInformation}(F) = -\sum_{R \in \mathbf{R}} \sum_{F' \in \mathcal{M}(F)} \log(P_R(F')).$$

where $P_R(F')$ signifies the probability that the entities in F' are related in R.

For the terms with $\mathbf{t}(F') = R$, the probability $P_R(F')$ can be computed directly using the MaxEnt model as $P_R(F') = P_R^{F'}(1)$, i.e. the probability that the relationship instance specified by the entities in F' is present in $\mathcal{R}_R$.

For the terms with $\mathbf{t}(F') \subset R$, the probability that there *exists* at least one relationship instance of type R involving all entities from F' is computed as $P_R(F') = 1 - \prod_{I=\{i_t\}_{t \in R \setminus \mathbf{t}(F')}} \left(1 - P_R^{F' \cup I}(1)\right)$. This follows directly from the independence of the different entries in the tensor $\mathbf{D}_R$, under the MaxEnt model.

We define the description length of a pattern $F \in \mathcal{F}_{\mathbb{D}}$ by specifying for each entity whether it does or it does not belong to the pattern. We use $-log(p)$ bits to specify that an entity belongs to the pattern and $-log(1-p)$ bits to specify that it does not, where p is a probability parameter. The description length of a pattern is given by the following equation:

$$\text{DescriptionLength}(F) = -\sum_{i \notin F} \log(1-p) - \sum_{i \in F} \log(p).$$

5 Experiments

In this section we present a qualitative evaluation of our method, by discussing some of the top ranked patterns on real-world data sets and a quantitative evaluation by showing scalability results and comparisons with other methods. For all the experiments we used parameter p equal to the density of the dataset.

5.1 Qualitative Evaluation

Here we show the highest ranked patterns found in real world datasets. We performed experiments on two real world datasets, one created from Bibsonomy [1]

(*bibsonomy-journals*) and another one created from IMDB[1](*imdb-3years*). Figure 2 depicts the diagrams of these datasets showing how different entity types are related as well as the number of instances for every relationship type and every entity type. To create the *bibsonomy-journals* dataset we selected the bibtex entries that corresponded to journal papers, along with their authors, users and tags. To create the *imdb-3years* dataset we created a view of the IMDB database containing person names and roles, as well as genres of movies produced from year 2008 to year 2010 and are not of the genre "Short". The role of a person in a movie can be actor, actress, producer, director etc.

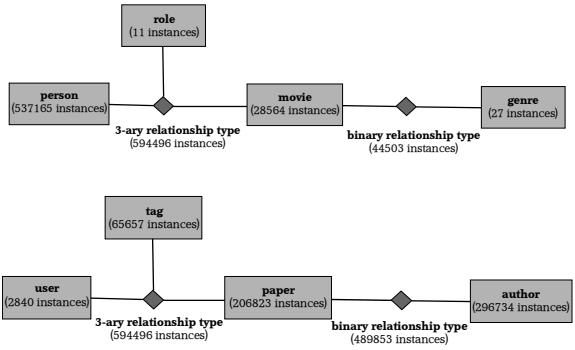

Fig. 2. Diagrams of the *imdb-3years* (top) and *bibsonomy-journals* datasets

We run experiments on both these datasets using different constraints. The results were ranked based on the interestingness measure we introduced in Sec. 4.2. In order to avoid redundancy we iteratively output the top ranked pattern and then re-rank the pattern set by taking into account in the interestingness of a pattern only relationship instances that have not been presented before.

The first ranked pattern on the *imdb-3years* dataset when requiring at least 2 papers, 2 titles, 2 roles and 1 genre, contains as entities: **2 persons:** Joel Coen; Ethan Coen; **4 titles:** The Yiddish Policemen's Union; True Grit; A Serious Man; Burn After Reading; **2 roles:** Director; Producer; **1 genre:** Drama. This pattern ranks first as it contains a lot of information (20 of relationship instances) conveyed in a concise way (9 entities). Recall here that our measure of self-information is an additive measure over the number of relationship instances and our measure of description length is an additive measure over the entities. The second pattern contains 18 relationship instances conveyed with 9 entities.

In the first experiment on *bibsonomy-journals* we required at least 2 entities from each entity type. The top pattern contains: **3 users:** 650753; 594907; 576800; **5 tags:** meningococcal; meningitidis; infections; neisseria; human; **6 papers:** (not shown here due to space constraints); **2 authors:** Heike Claus; Matthias Frosch. This pattern ranks highly as it conveys a lot of information

[1] See `http://www.imdb.com/`

(102 relationship instances) in a concise way (16 entities). The second pattern, in comparison, contains 48 relationship instances and 12 entities. We also run an experiment not necessarily requiring any entities of the entity type *user*, but at the same time requiring at least 5 tags, 5 papers and 2 authors. The most interesting pattern remains the same as in the previous experiment.

5.2 Quantitative Evaluation

In this subsection we show the performance of N-RMiner by presenting scalability results as well as results comparing N-RMiner with other algorithms on special cases of multi-relational data. All the experiments were run on a CentOS Linux machine with Intel Xeon@2.6GHz and 24GB of RAM.

In order to test the scalability we used the *bibsonomy-journals* dataset, of which we got five snapshots of increasing size. The first four snapshots were produced by randomly picking 0.01%, 0.1%, 1% and 10% of the entities of the type "paper" and selecting the entities of the other types that correspond to them. The final snapshot contains all of the *bibsonomy-journals* dataset. The results are shown in Table 1. The "-"s mean that the particular run did not finish within 2 days. As with most local pattern mining algorithms, time scales exponentially to the input size. However, when using constraints of at least 2 entities per entity type N-RMiner runs within a few hours for data sizes of approximately 5×10^5. The space that N-RMiner consumes also increases exponentially to the input size. However it only grows up to 1.5 GB for the largest dataset. As we are unaware of any other local pattern mining method that considers the general case of multi-relational data, where there can be any number of relationships of any arity, we compare N-RMiner with methods for specific cases of multi-relational

Table 1. Scalability testing of N-RMiner for increasing subsets of *bibsonomy-journals*

Constraints	#Entities	#N-MCCSs	Time (sec)	Space (MB)	Depth	Size of max N-MCCS
(0,0,0,0)	63	17	0	1.42	3	13
	1,167	295	2.17	3.63	5	100
	9,643	4,177	2599.22	28.32	14	800
	77,452	-	-	-	-	-
	567,416	-	-	-	-	-
(1,1,1,1)	63	12	0	1.40	1	13
	1,167	206	0.11	3.24	4	100
	9,643	2,186	8.52	18.99	5	83
	77,452	22,267	1,055.14	175.68	11	107
	567,416	-	-	-	-	-
(2,2,2,2)	63	0	0	1.37	0	0
	1,167	0	0.01	3.01	0	0
	9,643	0	0.6	16.98	0	0
	77,452	3	118.36	151.25	5	14
	567,416	359	13,121.4	1,502.95	16	27

data, namely Data-Peeler that mines n-Sets in one n-ary relationship ($n > 2$) [3] and RMiner which mines MCCSs in multi-relational data with binary relationships [14]. As we have shown in Sec. 2.3, both these pattern syntaxes constitute special cases of N-MCCSs and therefore the comparison of the algorithms is fair.

Table 2 (left) shows the time comparison between N-RMiner and Data-Peeler on random snapshots of increasing sizes of the 3-ary relationship of the *bibsonomy-journals* dataset. Data-Peeler offers the opportunity to use constraints on the minimum number of entities per entity type, like ours. However n-Sets are not as flexible as N-MCCSs in the sense that all entity types of the relationship need to be present in a pattern. We therefore compare the two algorithms in the simplest common setting of requiring at least one entity per entity type. The results show that N-RMiner outperforms Data-Peeler by at least one order of magnitude most of the times. Also for the dataset of 36,367 entities Data-Peeler could not run as it was crashing due to high memory requirements. This is because Data-Peeler uses a dense representation to store the dataset.

Table 2. Comparing N-RMiner w. Data-Peeler (left) and N-RMiner w. RMiner (right).

Cons-traints	#Entities	N-RMiner time (sec)	Data-Peeler time (sec)
(1,1,1)	39	0	0
	774	0.28	2.39
	5,405	97.75	2960.73
	36,367	182,156	-

Cons-traints	#Entities	N-RMiner time (sec)	RMiner time (sec)
(1,1,1)	3,291	2.81	1.65
	8,686	21.41	12.01
	51,203	1,296	610
	111,320	7,456	2,813

Table 2 (right) compares running times between N-RMiner and RMiner on increasing subsets from IMDB containing two binary relationships: between directors and titles, and between titles and genres. Since both algorithms use the same algorithmic framework [2] and the closure operator of N-RMiner is a generalisation of RMiner's (see Sec 3.2), it should be no surprise that they are similarly efficient, with a small slowdown of N-RMiner owing to its greater generality.

6 Conclusion

In this paper we took an important step towards the development of local pattern mining algorithms that can be directly applied to data as it often presents itself in real-life: represented in a relational database. More specifically, the N-MCCS pattern syntax and associated mining algorithm N-RMiner are applicable to multi-relational data with categorical attributes and n-ary relationships. Our approach is a strict generalisation of two less general problem settings [13,14,3], and it matches or even outperforms algorithms that were designed for these special cases. Additionally, we introduced a subjective interestingness measure for ranking the N-MCCSs found. An interesting research direction further increasing generality would be the extension of our results to ordinal and numerical data.

Acknowledgements. We thank L. Cerf for his help in running Data-Peeler. This work is funded by the grant EP/G056447/1 and by the PASCAL2 NoE.

References

1. Knowledge and data engineering group, University of Kassel: Benchmark folksonomy data from bibsonomy, version of January 1 (2012)
2. Boley, M., Horvath, T., Poigné, A., Wrobel, S.: Listing closed sets of strongly accessible set systems with applications to data mining. Theoretical Computer Science 411(3), 691–700 (2010)
3. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.-F.: Closed patterns meet n-ary relations. ACM Trans. Knowl. Discov. Data 3(1), 3:1–3:36 (2009)
4. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Mining and Knowledge Discovery 23(3), 407–446 (2011)
5. Dehaspe, L., Toivonen, H.: Discovery of frequent datalog patterns. Data Mining and Knowledge Discovery 3, 7–36 (1999)
6. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. Addison Wesley (2006)
7. Goethals, B., Page, W.L., Mampaey, M.: Mining interesting sets and rules in relational databases. In: Proc. of the ACM Symposium on Applied Computing (SAC), pp. 997–1001 (2010)
8. Jaschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Trias–an algorithm for mining iceberg tri-lattices. In: Proc. of the Sixth International Conference on Data Mining (ICDM), pp. 907–911 (2006)
9. Ji, L., Tan, K.-L., Tung, A.K.H.: Mining frequent closed cubes in 3d datasets. In: Proc. of the 32nd International Conference on Very Large Data Bases (VLDB), pp. 811–822 (2006)
10. Koopman, A., Siebes, A.: Discovering relational item sets efficiently. In: Proc. of the SIAM Conference on Data Mining (SDM), pp. 108–119 (2008)
11. Ng, E.K.K., Ng, K., Fu, A.W.-C., Wang, K.: Mining association rules from stars. In: Proc. of the 2002 IEEE Int. Conference on Data Mining, ICDM, pp. 322–329 (2002)
12. Nijssen, S., Kok, J.N.: Efficient frequent query discovery in FARMER. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS(LNAI), vol. 2838, pp. 350–362. Springer, Heidelberg (2003)
13. Spyropoulou, E., De Bie, T.: Interesting multi-relational patterns. In: Proc. of the IEEE International Conference on Data Mining, ICDM, pp. 675–684 (2011)
14. Spyropoulou, E., De Bie, T., Boley, M.: Mining interesting patterns in multirelational data. Data Mining and Knowledge Discovery (in print, 2013)