Bottom-up anytime discovery of generalised multimodal graph patterns for knowledge graphs [1]

Xander Wilcke [a,*], Rick Mourits [b], Auke Rijpma [c] and Richard Zijdeman [d]

[a] *Dept. of Computer Science, Vrije Universiteit Amsterdam, The Netherlands*
E-mail: w.x.wilcke@vu.nl

[b] *Data & Augmentation, International Institute for Social History, The Netherlands*
E-mail: rick.mourits@iisg.knaw.nl

[c] *Economic and Social History, Utrecht University, The Netherlands*
E-mail: a.rijpma@uu.nl

[d] *Data & Augmentation, International Institute for Social History, The Netherlands*
University of Stirling, Scotland UK
E-mail: richard.zijdeman@iisg.knaw.nl

Abstract. Vast amounts of heterogeneous knowledge are becoming publicly available in the form of knowledge graphs, often linking multiple sources of data that have never been together before, and thereby enabling scholars to answer many new research questions. It is often not known beforehand, however, which questions the data might have the answers to, potentially leaving many interesting and novel insights to remain undiscovered. To support scholars during this scientific workflow, we introduce an *anytime* algorithm for the bottom-up discovery of generalised multimodal graph patterns in knowledge graphs. Each pattern is a conjunction of binary statements with (data-) type variables, constants, and/or value patterns. Upon discovery, the patterns are converted to SPARQL queries and presented in an interactive facet browser together with metadata and provenance information, enabling scholars to explore, analyse, and share queries. We evaluate our method from a user perspective, with the help of domain experts in the humanities.

Keywords: Pattern Mining, Generalised Graph Patterns, Knowledge Graphs, Hypothesis Generation, Multimodal

1. Introduction

In only a short span of time, knowledge graphs have transitioned from an academic curiosity to an attractive data model for storing and publishing scientific data [20]. Amongst the multitude of adopters of this data model are many of the world's galleries, libraries, archival institutions, and museums [8, 17, 50], as well as various scientific communities including linguistics, archaeology, humanities, and history [9, 31, 33]. The combined efforts of these institutes and communities have resulted in a considerable number of publicly-available knowledge graphs which, together, surmount to vast amounts of interconnected heterogeneous knowledge. Much of this knowledge used to be stored in analogue or digital silos, and has never been brought together before. Now linked to one another, this

[1] This research was funded by CLARIAH-PLUS (NWO Grant 184.034.023)

[*] Corresponding author. E-mail: w.x.wilcke@vu.nl.

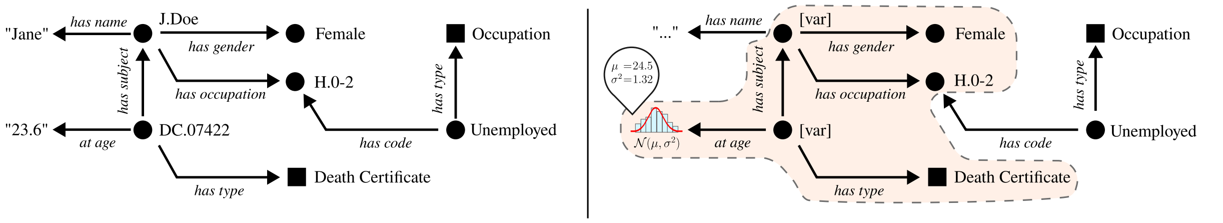


Figure 1. An example of a subgraph in the civil registry domain (left) and a possible graph pattern (right). Circles and squares represent entities and classes, respectively, and attribute values are within quotation marks. The shaded area indicates the structural component of the pattern, whereas the distribution conveys the non-structural component.

federative network of knowledge offers great opportunities for scholars, who can now potentially ask and answer many new research questions.

Drafting research questions is an essential step in the scientific research workflow. Such questions can either be derived from the scholarly literature or from patterns in data. However, without study, it is often not known beforehand which questions these data might have the answers to. Even if accompanied by rich metadata, these alone are often not enough to guide scholars in this process, limiting them to insights from the literature or sparks of their own imagination. This may result in many potentially interesting and novel insights to remain unstudied, due to possible biases or blind spots in the literature and the scholars' thinking. This work aims to support scholars during this early stage of the scientific workflow, by highlighting potentially interesting patterns in their data that may form the building blocks for new research questions, and which can be used as evidence for already existing lines of research.

Pattern detection on graph-shaped data can take on various forms. On the most fundamental level, graph patterns are recurrent and statistically significant subgraphs in which some or all of the vertices have been replaced by unbound variables [26]. Generalised graph patterns take this a step further, by having special variables that cover an entire set of vertices, such as all members of a certain class [21]. Scholars can use such patterns to explore *structural* regularities in the graph; other regularities, such as those between the various numerical, temporal, and textual attributes values are generally not considered, however, despite their prevalence in many knowledge graphs. This is particularly evident in the soft sciences where measurements, dating, and note taking are commonplace [48]. Since these multimodal data often contain insightful and unique information about the subject they belong to, it becomes all the more important to treat them as first-class citizen. By doing so, we can integrate non-structural regularities into generalised graph patterns and obtain more expressive patterns that offers scholars a more fine-grained view of their data.

Figure 1 illustrates the merit of combining structural and non-structural regularities. The left side of the figure depicts a civil record about an unemployed woman, named Jane, who died at an age of 23.6 years old, whereas, on the right, a graph pattern is shown that covers this record. The depicted pattern likewise covers other records about unemployed women, provided that they died at an age that falls within the learned distribution[2]. This distribution is an example of a non-structural regularity; without it, the pattern would have been limited to unemployed women whose age of death is on record, irrespective of the value. For other attributes, in this example the person's name, the variation between values might be too great to constitute a regularity, hence it being excluded from the pattern.

In this work, we introduce an algorithm for the *bottom-up* discovery of generalised multimodal graph patterns in knowledge graphs. The patterns are generated directly from the graph, leveraging both statistics and semantics to guide the discovery process, and get more precise with each following iteration. This gives our algorithm the *anytime* property, as scholars can terminate the process at their leisure and still obtain potentially interesting, albeit more generic, patterns. Additionally, special attention is given to the multimodal nature of many knowledge graphs, by allowing the combination of structural regularities with those between numerical, temporal, and textual attributes.

[2] We maintain $\mu \pm \sigma$ as the range for a match.

Moreover, to mitigate the curse of dimensionality, our algorithm incorporates smart pruning strategies and other optimization techniques.

We evaluate our method and the patterns it yields from a user perspective, by generating graph patterns from historical civil registry data and by assessing these together with the community of Northwest European social and economic historians. To lessen the semantic gap, the patterns are automatically converted to SPARQL queries upon discovery, improving their interpretability and familiarity by users. As a final step, the queries are presented in an interactive facet browser together with metadata, provenance information, and graph visualizations, enabling scholars to explore and analyse the patterns, as well as reproduce and share relevant data selections.

To summarize, our main contributions are a) a novel *anytime* algorithm for the *bottom-up* discovery of generalised multimodal graph patterns in knowledge graphs, b) the natural integration of various non-structural regularities into generalised graph patterns, and c) an extensive evaluation together with domain experts.

2. Related work

Knowledge graph exploration methods aim towards increasing knowledge utility and range from exploratory search and analytics to summarization and profiling strategies [28]. Pattern detection methods belong to this latter group of exploratory strategies and generally fall in one of two categories: the symbolic approaches, which employ some form of (logical) rule mining, and the non-symbolic approaches, which often involve (graph) neural networks in an unsupervised learning setting. The method described in this paper falls into the first category, and is specifically chosen for its explainability and deterministic nature.

Rule mining for graphs

A considerable body of literature is available on rule mining for relation data. Many of these approach the problem from either an *inductive* or *frequentist* school of thought. Methods that belong to the former (e.g. [36, 37, 45]) generally apply *inductive logic programming*, which involves learning logical rules that explain all true arcs and no false ones [35]. This technique is less suited for knowledge graphs, however, due to challenges with scalability and the need for a closed world [38]. In contrast, frequentist approaches emphasize (relative) coverage, and typically involves the mining of association rules or frequent substructures. This work falls under the umbrella of frequentist approaches, by introducing a method to discover statistically relevant substructures in knowledge graphs.

Frequent subgraph mining is a mature field of research which involves finding all subgraphs that occur more than some predefined number of times [22]. These subgraphs, and the graphs from which they are mined, are assumed to be labelled simple graphs. For knowledge graphs, in which the vertices and arcs have types, it is therefore more interesting to look for *generalised* subgraphs, for example by abstracting away to the level of the classes [4, 30, 42], or by generalising over controlled vocabularies and taxonomies [6, 43]. To discover the subgraphs, these methods commonly employ a bottom-up approach, similar to our method, that begins with the most basic rules and incrementally adds new arcs until some condition is reached or the search space has been exhausted.

Closely related to subgraph discovery is graph-based association rule mining, which aims at finding rules that imply frequent co-occurrences between subgraphs. Such rules can be found by adapting the Apriori algorithm for graph data, in which case so-called *item sets* of correlated arcs are sought, which are then clustered hierarchically to induce an ordering on frequency [3, 46, 52]. Other methods are specifically tailored to graphs and use a bottom-up approach comparable to many subgraph mining methods [12, 32, 53]. In some cases, background knowledge is levered to infer implicit knowledge [34, 39].

Many methods employ optimization and smart pruning strategies to reduce the search space to a more manageable size, for example by cutting unviable branches at an early stage or by avoiding duplicate subgraphs found via different paths [13, 27, 53]. Similar strategies are being used by our method.

2.1. *Knowledge Graph Patterns*

Graph patterns have been explored in various different forms. Traditionally used in relational databases, *graph functional dependencies* express that entities with the same values for all attributes in one set must also have the same values for those in another set [1, 7, 19]. Follow-up work has extended these with paths, allowing for patterns over predicate-object pairs [18, 55], whereas more recent efforts explored the discovery of graph motifs with support for entities, literals, and variables [10, 11]. In [55], the authors include support for patterns over numerical attributes by clustering values using *k*-means. Our method likewise supports numerical patterns but instead fits statistical distributions over (subsets of) the value space.

Another form of graph patterns are so-called *knowledge patterns* [14], an adaption of *frames* to knowledge graphs which offer a generic framework for capturing knowledge. *Encyclopedic knowledge patterns* extend this notion by grounding the patterns in general-purpose knowledge bases [40, 41]. Moreover, the authors demonstrate how patterns can be extracted by querying the database. The discovered patterns are then evaluated via a user study, similar to our approach.

Query generation

To the best of our knowledge, generating queries directly from the data has received little attention. Instead, most literature explores query log mining [56], top-down query construction [16], or the use of language models [51] for this purpose. A notable exception is *F. Shen et al* [49], who cluster biomedical data on semantic closeness of the relationships and convert these clusters into SPARQL queries. Some studies have also looked into the conversion of SPARQL queries into other formats, including logical formulae [44, 47]. Our approach performs a similar transformation, but in the other direction: from formulae to queries.

3. Prerequisites

Central to our approach are knowledge graphs and SPARQL queries. This next section will briefly introduce these concepts.

3.1. Knowledge graphs

A knowledge graph $G = (\mathcal{R}, \mathcal{P}, \mathcal{A})$ is a labelled multidigraph with $\mathcal{R}$ and $\mathcal{P}$ denoting the set of resources (vertices) and predicates (arc types), respectively, and with $\mathcal{A} \subseteq \mathcal{P} \times \mathcal{E} \times \mathcal{E} \cup \mathcal{P} \times \mathcal{E} \times \mathcal{L}$ representing the set of all assertions (arcs) that make up the graph. The set of resources $\mathcal{R} = \mathcal{E} \cup \mathcal{L}$ can be further divided into the set of entities, $\mathcal{E}$, which represent unique things, tangible or otherwise, and the set of literals, $\mathcal{L}$, which represent attribute values such as text and numbers, and which belong to exactly one entity. Literals can optionally be annotated with their datatype (or language tag, from which the datatype can be inferred) which itself is an entity.

An example of a knowledge graph is depicted in Figure 1-left, showing a small graph from the civil registry domain. This particular graph contains seven entities, two of which are classes, and two literals: a number and a string. These elements are linked to each other by exactly eight assertions, two of which represent the same predicate: *has_type*.

There are various data models available to model knowledge graphs with. In this work, we consider the *Resource Description Framework* (RDF)[3], which is a popular choice for this purpose. However, our approach can be adapted to other data models with minor changes.

3.2. SPARQL queries

SPARQL[4]is a query language for RDF-encoded knowledge graphs that supports searching for graph patterns. A typical SPARQL query consists of three parts: 1) a prologue, in which the namespaces are defined, 2) a `SELECT` clause, which specifies the return variables, and 3) a `WHERE` clause, which contains the graph pattern we are to match against. SPARQL also provides many other capabilities, but these are out of the scope of this paper.

[3] The RDF specification is available at www.w3.org/TR/rdf11-concepts

[4] The SPARQL specification is available at www.w3.org/TR/sparql11-query

Graph patterns in a SPARQL query are similar to their logical counterpart except that the clauses are written in infix notation—$\mathcal{R} \times \mathcal{P} \times \mathcal{R}$—and that the conjunctions between them are implicit. Additionally, variables are prepended by a question mark (`?`), and the `FILTER` keyword can be used to constrain the result set. An example is listed in Listing 1-right, showing the SPARQL query corresponding to the graph pattern in Figure 1-right.

4. Defining generalised multimodal graph patterns

Generalised multimodal graph patterns are recurrent and statistically significant subgraphs in which some or all of the resources have been replaced by special variables. This allows for graph patterns that abstract away from the level of the individual resources by modelling structural regularities between and non-structural regularities within sets of resources. From now on, we will refer to such patterns as graph patterns or simply as patterns unless the meaning is not evident from the context.

Formally, a graph pattern $\phi = c_i \wedge c_j \wedge \ldots c_k$ is a conjunction of k clauses, with $k \geqslant 1$, where each clause $c = p(a, b)$ is a binary predicate that represents the relationship $p \in \mathcal{P}$ between the elements a and b. Here, a and b can be constants that represent actual resources in the graph, in which case $p(a, b)$ corresponds to an assertion in $\mathcal{A}$, or they can be variables, representing a set of entities or literals. In either case, we will refer to a and b as the *head* and *tail* of a relationship, respectively.

In this work we consider three different kinds of variables, namely *object-type*, *data-type*, and *value-range* variables. The set of resources that each variable covers is called its *domain*. For each of the three variable types, we define their domain as follows.

Object-type: *Let $\mathcal{T}_\mathcal{E}$ be the set of object types in G, and $T(e, t)$ a binary predicate that holds if entity $e \in \mathcal{E}$ is of type t. The domain of an object-type variable of type $t \in \mathcal{T}_\mathcal{E}$ can now be defined as the set of entities $\mathcal{E}_t \subseteq \mathcal{E}$ such that $\forall e \in \mathcal{E}_t : T(e, t)$.*

Data-type: *Let $\mathcal{T}_\mathcal{L}$ be the set of datatypes in G, and $T(\ell, t)$ a binary predicate that holds if literal $\ell \in \mathcal{L}$ is of datatype t. The domain of a data-type variable of type $t \in \mathcal{T}_\mathcal{L}$ can now be defined as the set of literals $\mathcal{L}_t \subseteq \mathcal{L}$ such that $\forall \ell \in \mathcal{L}_t : T(\ell, t)$.*

Value-range: *Let predicate $p \in P$ represent a relationship with value space $\mathcal{S} \subseteq \mathcal{L}$ such that $\forall \ell \in \mathcal{L}, \exists e \in \mathcal{E} : p(e, \ell) \implies \ell \in \mathcal{S}$. The domain of a value-range variable can now be defined as the set of attribute values $\mathcal{S}_F \subseteq \mathcal{S}$ that fall within a distribution F defined on $\mathcal{S}$.*

Both object-type and data-type variables allow for a generalisation over structure. Examples of the former are the object types `Person` and `Occupation`, which cover all people and jobs, whereas the datatypes `String` and `Float` encompass all text and real-valued attribute values. Value-range variables offer a further generalisation over attribute values, for example by fitting one or more Gaussian distributions on a collection of years, or by defining a uniform distribution over a set of characters (encoded as regular expression, e.g. `"^[:alnum:]{3,6}$"`).

The clauses in a pattern are subject to several rules to safeguard their logical and semantic validity. Firstly, the head of a clause *must* be an object-type variable, for else the pattern is bound to a specific resource, making generalisation impossible. Secondly, for all-but-one object-type variables in the head of a clause there *must* exist a clause which has the same variable in the tail position, thus ensuring a connected graph pattern. Third and final, the tail of a non-terminal clause *must* be an object-type variable: ending such as clause with a data-type variable, a value-range variable, or a literal is semantically invalid, whereas ending it with an entity is nonsensical since any continuation from that point onwards will not result in a reduction of the pattern's domain. In contrast, the tail of a *terminal* clause can be a resource or any kind of variable.

We organize graph patterns based on depth, length, width, and support. The depth of a pattern equals the longest path between any two elements, whereas the length and width equal the number of clauses in total and the maximum number of clauses with the same head, respectively. The support value equals the number of occurrences of a pattern in a specific dataset. An example graph pattern is depicted in Figure 1-right, which has a depth of four hops, a length and width of five and three clauses, respectively, and with an unknown support value. The logical equivalent of this pattern is listed in Listing 1-left.

$$
\begin{aligned}
\phi = {} & \mathit{has\_gender}(v_i, \texttt{Female}) \\
& \wedge \mathit{has\_occupation}(v_i, \texttt{H.0-2}) \\
& \wedge \mathit{has\_subject}(v_j, v_i) \\
& \wedge \mathit{has\_type}(v_j, \texttt{Death\_Certificate}) \\
& \wedge \mathit{at\_age}(v_j, \mathcal{N}(24.5, 1.32))
\end{aligned}
$$

```
SELECT ?v_i ?v_j
WHERE {
    ?v_i has_gender Female .
    ?v_i has_occupation H.0-2 .

    ?v_j has_subject ?v_i .
    ?v_j has_type Death_Certificate .
    ?v_j at_age ?v_k .

    FILTER (
           ?v_k >= "23.35"^^int
        && ?v_k <= "25.65"^^int
    )
}
```

Listing 1: The graph pattern from Fig. 1-right in logical notation (left) and as SPARQL query (right). Variables v_i and v_j correspond to the two unbound resources in the figure. Note that, for brevity, the namespaces have been omitted.

5. Discovering graph patterns

Our algorithm employs a two-phase approach for discovering graph patterns. During the first phase, the algorithm generates all possible single-clause patterns that satisfy the minimal requested support. These so-called *base patterns* form the building blocks for more complex graph patterns, which are generated during the second phase by extending previously discovered graph patterns with appropriate base patterns. Since all complex graph patterns are a combination of base patterns, and since generating and evaluating new patterns involve simple set operations, minimal further resource-intensive computation is necessary after completing the first phase. By also providing each pattern with a description of its domain (e.g. via a set of integer-encoded resources) we no longer require to keep the original graph in memory while retaining the minimal information necessary to derive the domain and support for new patterns.

New graph patterns are generated *breadth first*, by first computing all possible patterns of minimal size and by then iteratively combining these to form ever more complex patterns. This gives our algorithm the *anytime* property, as users can terminate a run at their leisure while still obtaining potentially-interesting, albeit less complex, results. Our algorithm is also *embarrassingly parallel*, as each new pattern effectively starts a separate branch which can be computed independent from any of the other branches.

Please note that, for the purpose of conciseness, all procedures shown are simplified by leaving out pruning points and other optimization techniques.

5.1. Constructing base patterns

Base patterns are generated by generalising over all assertions of which the entity in the head position is of the same type, as shown in Procedure 1. This type-centric approach is chosen because the members of a class are likely to possess similar characteristics and, by extension, are also likely to share similar regularities. By replacing the specific head entities in these assertions by their corresponding object-type variables, we obtain clauses of the form $p(\upsilon^t_{ot}, r)$ which represent a relationship p between an entity of type t and a resource r. After computing the domain and support, each clause that enjoys a sufficiently high score is made into a pattern, $\phi = p(\upsilon^t_{ot}, r)$, and added to polytree Ω as root.

For brevity, the pseudocode in Procedure 1 omits the computation of clauses with a variable in the tail position. Similar steps can be used, however, to generate the remaining three cases: for object-type and data-type variables, we simply need to keep count of the various types of entities and literals, respectively, and, when this count meets the minimal requested support, create a new base pattern $\phi = p(\upsilon^t_{ot}, \upsilon^{t'}_{ot})$ or $\phi = p(\upsilon^t_{ot}, \upsilon^{t'}_{dt})$, with $\upsilon^{t'}_{dt}$ a data-type variable of type t', which then gets added to Ω. For value-range variables, however, a few additional steps are needed.

Procedure 1 The procedure (simplified) for computing all base patterns with a minimal support value. Only the case with a single object-type variable (υ_{ot}^{t}) is shown (line 12); the other cases, which have variables on both sides of the clause, are similar but require an extra step to calculate the domain and/or range.

```
 1: function ComputeBasePatterns($G$, $supp_{min}$)
 2:     $\Omega :=$ empty list
 3:     $\Omega.addItem($empty map$)$
 4:     for type $t$ in $\{t \mid \exists e \in \mathcal{E} : type(e,t)\}$ do
 5:         $\mathcal{B} :=$ empty set
 6:         $\mathcal{E}_t := \{e \in \mathcal{E} \mid type(e,t)\}$
 7:         if $|\mathcal{E}_t| \geqslant supp_{min}$ then
 8:             for $p \in \mathcal{P}$ do
 9:                 $\mathcal{U} := \{p(e,r) \mid \exists e \in \mathcal{E}_t, \exists r \in \mathcal{R} : p(e,r) \in \mathcal{A}\}$
10:                 for $p(\cdot,r) \in \mathcal{U}$ do ‘
11:                     if $|p(\cdot,r) \in \mathcal{U}| \geqslant supp_{min}$ then
12:                         $\phi := p(\upsilon_{ot}^{t}, r)$
13:                         $\mathcal{B} := \mathcal{B} \cup \{\phi\}$
14:         $\Omega(0,t) := \mathcal{B}$
15:     return $\Omega$
```

Procedure 2 The procedure (simplified) to iteratively discover more complex graph patterns, by matching possible endpoints (line 9) with appropriate base patterns (line 11). Function $\Delta(\cdot)$ returns the depth of an element.

```
 1: function Discover($G$, $supp_{min}$, $d_{max}$)
 2:     $\Omega :=$ ComputeBasePatterns$(G, supp_{min})$

 3:     $d := 0$
 4:     while $d < d_{max}$ do
 5:         for type $t$ in $\Omega.types()$ do
 6:             $\mathcal{O} :=$ empty set
 7:             for $\phi \in \Omega(d,t)$ do
 8:                 $\mathcal{C} :=$ empty set
 9:                 $\mathcal{I} := \{c = p(\cdot, \upsilon_{ot}^{t}) \mid c \in \phi \wedge \Delta(\upsilon_{ot}^{t}) = d\}$
10:                 for $c_i = p_k(\cdot, \upsilon_{ot}^{t}) \in \mathcal{I}$ do
11:                     $\mathcal{J} := \{c = p(\upsilon_{ot}^{t}, \cdot) \mid c \in \Omega(0,t)\}$
12:                     for $c_j = p_l(\upsilon_{ot}^{t}, \cdot) \in \mathcal{J}$ do
13:                         $\mathcal{C} := \mathcal{C} \cup \{(c_i, c_j)\}$
14:                 $\mathcal{O} := \mathcal{O} \cup$ Explore$(\phi, \mathcal{C}, supp_{min})$
15:             $\Omega(d+1,t) := \mathcal{O}$
16:         $d := d + 1$
17:     return $\Omega$
```

Value-range variables are generated by defining one or more distributions over all the literal values that occur on the right-hand side of a relationship p with entities of type t. For numerical data, this involves fitting multiple Gaussian mixture models with various seeds and different number of modes, and by then evaluating these fits using the Bayesian Information Criterion (BIC). For temporal data, such as dates, months, and durations, the same procedure is followed but now the values are first converted into seconds (Unix time). Additionally, in either case the values are standardized, shuffled, and augmented with a tiny amount of Gaussian noise to improve the fit. The fitted distributions $F_1, F_2, \ldots, F_n$ are made into value-range patterns $p(\upsilon_{ot}^{t}, \upsilon_{vr}^{F_i})$ if the number of literal values they cover meets the minimal requested support.

Procedure 3 The produce (simplified) to evaluate the candidate extensions, and all legal combinations thereof. Function `supp(·)` returns the support value for a given pattern.

```
function EXPLORE($\phi$, $\mathcal{C}$, $supp_{min}$)
    $\mathcal{O} :=$ empty set
    $Q :=$ empty queue

    $Q.enqueue(\phi)$
    while $Q \neq \emptyset$ do
        $\psi := Q.dequeue()$
        for $c_i, c_j \in \mathcal{C}$ do
            $\psi' := \psi \wedge c_j$                    ▷ $\psi = c_1 \wedge c_2 \wedge \ldots \wedge c_i$
            if supp($\psi'$) $\geqslant supp_{min}$
            then
                $\mathcal{O} := \mathcal{O} \cup \{\psi'\}$
                $Q.enqueue(\psi')$
    return $\mathcal{O}$
```

The final variant of a value-range variable targets textual data, including natural language and arbitrary strings, and involves the generation of hierarchical regular expressions. This is accomplished by first generating regular expressions for each value separately, clustering these by similarity, and by then generalising the expressions until they cover (almost) all members. We align these expressions with our earlier definition of a domain by regarding them as uniform distributions to specific character sets.

5.2. Combining graph patterns

Going from the base patterns to more complex patterns involves the generation of candidate extensions $\mathcal{C}$, which, if deemed favourable, are appended to their parents' set of clauses to form new graph patterns ψ'. These new patterns are then added to Ω as children to their parents, provided that they meet the minimal requested support. Procedure 2 and 3 outline this process.

Each of the candidate extensions is a pair of clauses (c_i, c_j), where $c_i = p_k(\cdot, v^t_{ot})$ is one of the parent's outer clauses—a candidate endpoint—and $c_j = p_l(v^t_{ot}, \cdot)$ is a suitable base pattern. Both clauses are ensured to hold the same object-type variable, thus providing a semantically valid connection. Depending on the element in the tail position of clause c_j, the extension, if added, will be terminal or non-terminal. If a pattern has no further non-terminal clauses it will be omitted from future iterations of the algorithm.

There are often multiple extensions possible per pattern within the same iteration. To exhaustively explore the space of multiple extensions, our algorithm evaluates each of these extensions separately, as well as their k-combination (without repetition) with k ranging from two to the number of candidate extensions $|\mathcal{C}|$. Note that, since not all combinations are accepted, the actual maximum value for k will generally be less than $|\mathcal{C}|$ in practice.

For each new pattern we compute the domain and associated support score. Since patterns carry a description of their own domain, we can easily compute the domain of a newly derived pattern by taking the intersection of its parent's domain with that of the recently-added clause, and by then propagating this change through the other clauses in the pattern. Figure 2 illustrates this principle, by showing how adding a new clause reduces the domain of the pattern as a whole.

5.3. Search optimization

Smart pruning techniques and other optimizations are used to reduce the search space by avoiding duplicate, invalid, and/or poorly supported patterns and clauses. We provide a brief description of the most important techniques next.

- Since every added clause makes a pattern more specific, it must follow that the corresponding domain should be a proper subset of that of its parent. Hence, patterns that have the same domain as their parent are pruned and disallowed from becoming a parent themselves. The sole exception are clauses with an object-type variable as tail, which are kept for one iteration more in case they might farther a pattern that *does* reduce the domain.
- Patterns that were not extended during the current iteration are omitted from future iterations. The intuition behind this is that future iterations necessarily involve more specific patterns; if the patterns did not meet the minimal support during the current iteration, then it follows that this will also be the case for future iterations. The same holds for base patterns.
- Candidate extensions that do not meet the minimal required support are omitted from future iterations. Since the domain of an extension will stay unchanged during the entirety of a run, it follows that adding them can never result in a pattern with a sufficiently high support score. Base patterns for which this is the case are already filtered during their creation.
- Duplicate patterns (which might occur via different routes) are pruned early on by creating a cheap proxy—the logical formula as string—and checking this against a hash table before creating the actual object and computing its domain.
- Patterns that only have terminal clauses or no appropriate object-type variables are disallowed from becoming a parent, whereas patterns which exceed the maximum allowed length, width, or depth are pruned early on for obvious reasons.

5.4. Pattern browser

A simple facet browser (Fig. 3) was created to assist scholars with the exploration and analysis of the discovered patterns during evaluation. This browser, named the *pattern browser*, facilitates the filtering of patterns over various dimensions, including *support*, *depth*, *length*, and *width*, as well as provide full-text search capabilities. The filtered selection can be saved in a separate file, which itself can be opened in the pattern browser for further analysis. Alternatively, the saved selection can be shared with others or published on the web, facilitation reuse and reproducibility.

Metadata is stored together with the patterns and can be viewed directly from the pattern browser. Amongst others, these data include provenance information and hyperparameter settings from the process that created the patterns. Additional data is appended to the provenance information upon saving a filtered selection, allowing users to trace back all performed actions. Both patterns and metadata are stored using RDF.

By relying on open web standards and structured, machine-readable metadata, and by encouraging reproducibility and sharing, the pattern browser embraces the philosophy behind FAIR principles and open science [54]. It is the authors' hope that this will aid in the long-term sustainability of the generated research output and that it will foster future collaborations between scholars working with graph patterns.

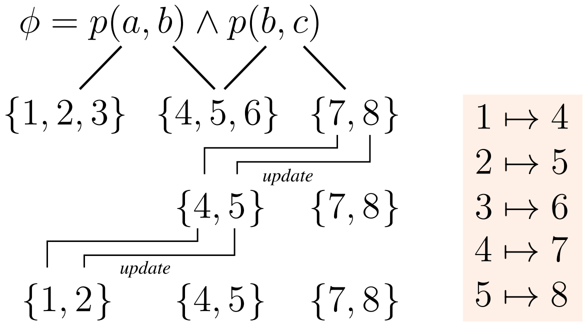
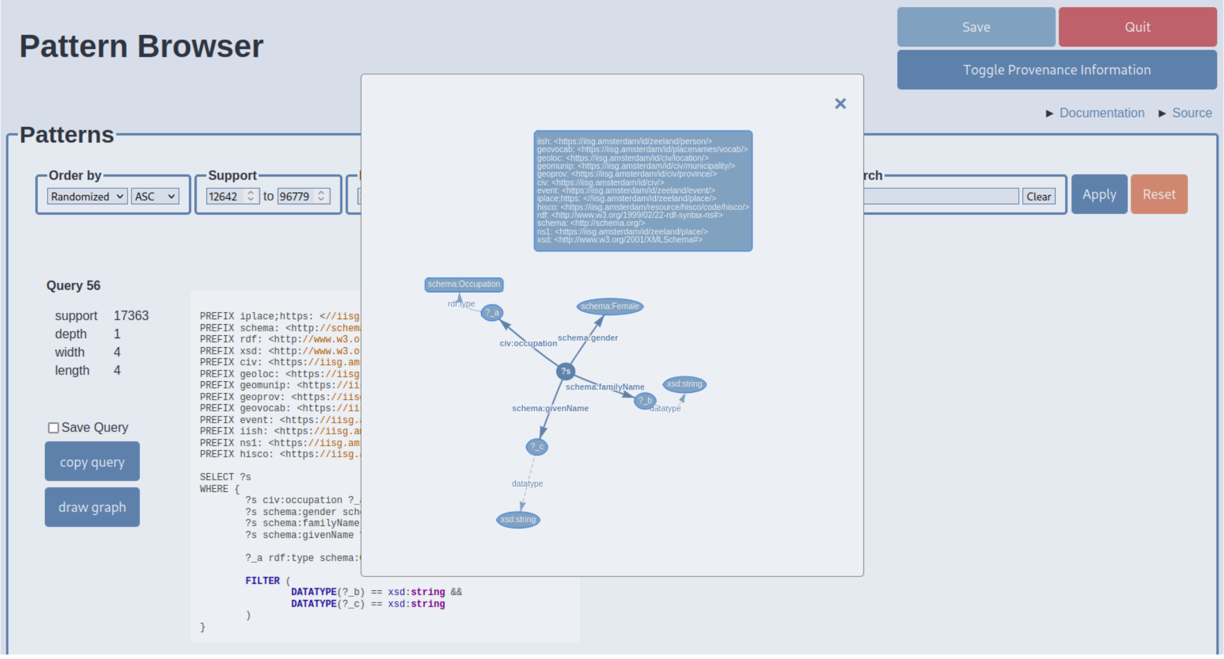


Figure 2. Updating the domain of a pattern ϕ after adding clause $p(b, c)$. Domains are depicted as sets with integer-encoded resources, whereas the maps between resources represent assertions in the graph (e.g., $p(1, 4)$). Since resource 6 is not connected to any of the resources in the domain of c, adding $p(b, c)$ thus reduces the domain of b (by removing resource 6), which, in turn, reduces that of a (by removing resource 3).



support	17363
depth	1
width	4
length	4

```
PREFIX iplace;https: <//iisg
PREFIX schema: <http://schem
PREFIX rdf: <http://www.w3.o
PREFIX xsd: <http://www.w3.o
PREFIX civ: <https://iisg.am
PREFIX geoloc: <https://iisg
PREFIX geomunip: <https://ii
PREFIX geoprov: <https://iis
PREFIX geovocab: <https://ii
PREFIX event: <https://iisg.
PREFIX iish: <https://iisg.a
PREFIX ns1: <https://iisg.am
PREFIX hisco: <https://iisg.

SELECT ?s
WHERE {
        ?s civ:occupation ?_
        ?s schema:gender sch
        ?s schema:familyName
        ?s schema:givenName

        ?_a rdf:type schema:

        FILTER (
                DATATYPE(?_b) == xsd:string &&
                DATATYPE(?_c) == xsd:string
        )
}
```

```
iish: <https://iisg.amsterdam/id/zeeland/person/>
geovocab: <https://iisg.amsterdam/id/placenames/vocab/>
geoloc: <https://iisg.amsterdam/id/civ/location/>
geomunip: <https://iisg.amsterdam/id/civ/municipality/>
geoprov: <https://iisg.amsterdam/id/civ/province/>
civ: <https://iisg.amsterdam/id/civ/>
event: <https://iisg.amsterdam/id/zeeland/event/>
iplace;https: <//iisg.amsterdam/id/zeeland/place/>
hisco: <https://iisg.amsterdam/resource/hisco/code/hisco/>
rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
schema: <http://schema.org/>
ns1: <https://iisg.amsterdam/id/zeeland/place/>
xsd: <http://www.w3.org/2001/XMLSchema#>
```

Figure 3. A screenshot of the facet browser showing a graph pattern encoded as SPARQL query and visualized as a graph.

6. Evaluation

We evaluate our algorithm and the patterns it produces from a user-centric perspective, by conducting a user study amongst a select group of domain experts from the humanities. The primary goal of this user study is to ascertain the perceived interestingness of the discovered patterns, as well their interpretability. For this purpose, graph patterns were discovered within a domain-specific knowledge graph and presented to experts to assess. The implementation of our algorithm that was used to generate these patterns is available online[5]. All runs of this algorithm were performed on the DAS-6 supercomputer [2].

6.1. Dataset

The knowledge graph used in our experiments contains the civil records from Dutch citizen who were alive between 1811 and 1974, and was created as part of the *LINKS* project [29] on reconstructing historical life courses. Each record in the dataset includes information about a person's pedigree, marital status, occupation, and location, as well as various important life events including birth, death, and becoming a parent. Due to the sensitivity of these data we are prohibited from sharing this dataset, unfortunately.

In its entirety, the dataset contains the records from over 5.5 million people. For experimental purposes, a subset was created by randomly sampling 100 thousand individuals together with their context, resulting in a graph with just over one million assertions between roughly 635 thousand resources. We believe that these numbers are sufficiently large enough for the same patterns to emerge as those present in the original dataset.

6.2. Exploratory Analysis

To generate graph patterns for our user study we ran our method with a minimal support value of 2500 and a maximum depth of three. These numbers were determined experimentally, by performing a grid search and by

[5] See gitlab.com/wxwilcke/hypodisc

```
SELECT ?v_i ?v_j
WHERE {
    ?v_i has_gender Male .
    ?v_i has_occupation H.61220 .
    ?v_i has_age ?v_k .

    ?v_j has_subject ?v_i .
    ?v_j has_type Mariage_Certificate .

    FILTER (
      ?v_k == "29"^^int
    )
}
```

"In this population sample, 1,238 out of 100,000 records are about 29 year old married men who work in agriculture."

```
SELECT ?v_i
WHERE {
    ?v_i has_gender Female .
    ?v_i has_firstName ?v_j .
    ?v_i has_familyName ?v_k .

    FILTER (
      REGEX(?v_j, "[a-z]{2,14}\s[a-z]{2,14}")
      && REGEX(?v_k, "[a-z]{3,16}")
    )
}
```

"In this population sample, 13,632 out of 100,000 records are about women with two first names between 2 and 14 characters each, and with a family name between 3 and 16 characters."

Listing 2: Two graph patterns that were discovered in the civil registry dataset, encoded as SPARQL queries, together with their natural language description. The patterns were selected because they highlight two distinctly different regularities. Note that, for brevity, the namespaces have been omitted.

observing the rate of pattern discovery over a short period. Using these hyperparameters, the entire run took roughly four hours to complete and yielded 720 candidate patterns.

A closer look at the candidate patterns reveals that most describe regularities within various groups of people, determined by a shared occupation, age, and/or event location. The large majority of these patterns have a support value between roughly 2600 and 7300, with a mode approximately of 3800. Lengthwise, the discovered pattern encompass between two and six assertions, with a length of four as the mode. These characteristics fall in line with our expectations given the dataset.

Two patterns that were found are listed as examples in Listing 2, together with their description in natural language. These examples were selected because they highlight two distinctly different regularities. More precisely, the left-hand pattern covers the set of all 29 year old men who are married and who work in the agricultural sector, which accounts for 1,238 individuals in the dataset. The graph pattern on the right accounts for 13,632 people, and encompasses all women with two first names between two and 14 characters each, and with a family name between three and 16 characters.

6.3. User study

The user study took the form of an online survey, lowering the barrier for participation and allowing for a cross-border audience. To ensure a good fit between this audience and the topic at hand it was decided to make the survey open to invitation only. While this might have resulted in a lower number of participants, we are confident that their responses are more valuable.

The survey was split into four sections. In the first section, participants were asked about their familiarity with the core concepts surrounding this research. The answers to these questions allowed us to weight the participants' responses on later questions. The second and third sections involved questions about the graph patterns and the pattern browser, respectively, whereas the last section asked several overarching questions about the perceived usefulness of our method and the patterns it yields. In all cases (save for open questions) the responses were recorded using a five-point Likert scale ranging from *Strongly Disagree* (negative) to *Strongly Agree* (positive).

To assess the graph patterns on interestingness, participants were presented with several hand-picked patterns in SPARQL format, and asked to rate each one on *novelty*, *validity*, and *utility* (all of which are dimension of interestingness [15]) as well as on *interpretability*. A similar setup was used for the pattern browser, but instead

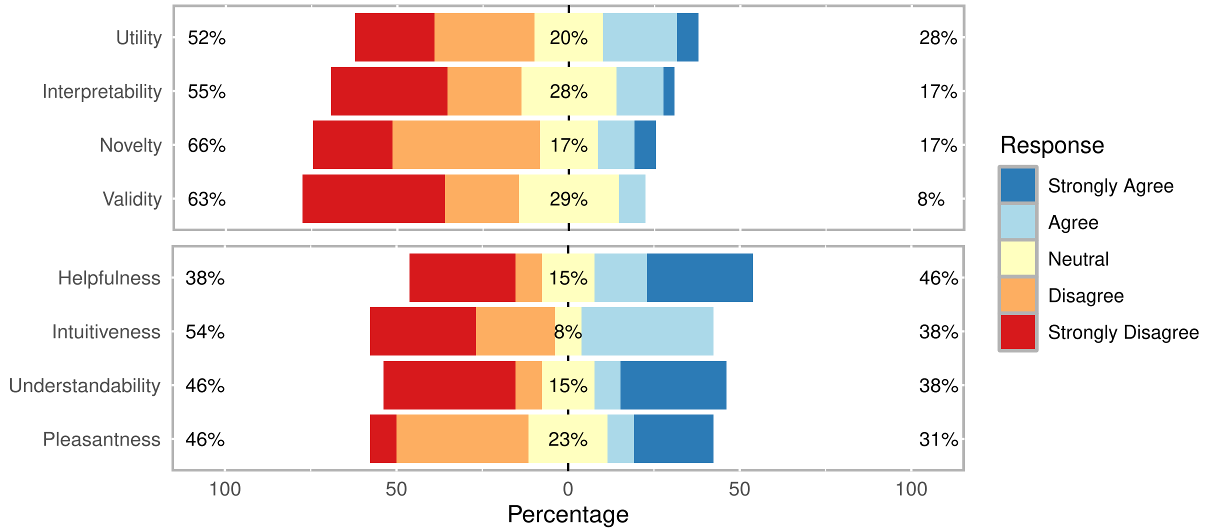


Figure 4. Responses (Kendall's $W = 0.14$) about the perceived novelty, validity, utility, and interpretability of the presented graph patterns (top) and the responses (Kendall's $W = 0.11$) about the perceived helpfulness, intuitiveness, understandability, and pleasantness of the pattern browser (bottom). All responses were measures on a 5-point Likert scale.

using screenshots and rated on *helpfulness* (in analysing the patterns), *intuitiveness* (of the interface), *pleasantness* (of the design), and *understandability* (of the displayed information).

To determine the agreement and reliability amongst participants, we employ Kendall's coefficient of concordance W for its suitability to evaluate ordinal data with multiple ratings over multiple items [24]. For similar reasons, we use Kendall rank correlation coefficient τ to measure dependencies between responses [23]. We furthermore employ factor analyses to obtain a better understanding of the interactions between criteria.

6.4. Results & discussion

The survey was distributed within the Northwest European community of social and economic historians, and specifically targeted scholars with experience in data science and *Semantic Web* technologies. A total of 13 out of the 42 scholars to whom we reached out took part in the user study, corresponding to a fair response rate of 31%. Table 1 lists their familiarity with the domain, knowledge graphs, SPARQL, and database terminology. The responses suggest that the participants only moderately align with the domain, as both the median and mode scores are *neutral*. Similar for their familiarity with knowledge graphs, albeit with a lower median of *disagree*. Even less familiar do the participants seem to be with SPARQL and database terminology, having a median of *disagree* and mode of *strongly disagree* for the former, and a median and mode of *disagree* for the latter. Together, these responses suggest a possible gap between the technical background and experience of the participants and the skills required to fully comprehend the method and the patterns it yields. While this discrepancy might not be evenly spread amongst the participants, as suggested by the relatively low agreement ($W = 0.29$), it may have induced a degree of uncertainty in the participants and in the answers they have provided. This is further corroborated by the self-reported confidence (Table 5), with roughly half of the participants (46%) giving themselves the lowest score ($W = 0.85$).

Figure 4-top shows the responses about the presented graph patterns. Overall, the provided scores suggest that the participants were critical about the discovered patterns, with 52% to 64% believing the patterns to be uninteresting against 8% to 28% deeming the opposite. However, the number of people who were very negative differ considerably from roughly one out of three to two out of three negative responses. This is supported by the low agreement ($W = 0.14$) amongst raters, which indicates a wide range of opinions. Looking at the underlying criteria, we observe that the participants seem the most positive (28%) about the utility of the patterns, followed by their novelty

Table 1

Familiarity of the participants (Kendall's $W = 0.29$) with the domain, knowledge graphs, SPARQL, and database terminology. Last column shows correlation (Kendall's τ) with perceived utility (Tab. 2).

Familiarity	Median	Mode	τ
Domain	*neutral*	*neutral*	-0.37
Graphs	*disagree*	*neutral*	0.42
SPARQL	*disagree*	*strongly disagree*	0.51
DB Terms	*disagree*	*disagree*	0.77

Table 2

Utility of the patterns (Kendall's $W = 0.54$) as perceived by participants in relative numbers.

Score	Portion
fully agree	0.00
agree	0.15
neutral	0.24
disagree	0.15
fully disagree	0.46

(17%). The patterns' validity, however, scores poorly with only few participants being positive (8%). This last score is particularly interesting since the patterns are generalisations of the original data, rather than predictions, and are therefore as valid as the data they are discovered on. That the patterns were nevertheless deemed invalid by most of the participants suggests that there are either problems with the chosen dataset (which is unlikely, it being a curated dataset) or that there was a mismatch between the experts' expectations and the output of our method. This latter reason seems more probable, since only few participants were positive about interpretability (17%).

An analysis of the factor loadings belonging to these responses (Table 3) shows a clear separation between criteria, with utility ($0.90\lambda_1$) and validity ($0.88\lambda_1$) on one hand, and interpretability ($0.97\lambda_2$) on the other. This suggests that both utility and validity contribute to the same latent component, which we can perhaps interpret as an indication of *effectiveness*, whereas interpretability measures an entirely different component of its own. Less clear cut is novelty, which enjoys significant cross loadings on both components ($-0.54\lambda_1 + 0.40\lambda_2$) which suggests that this criterion is a poor indicator for the dimensions on which the participants assess the usefulness of the patterns. Rather, novelty appears to be a combination of low effectiveness and high interpretability, suggesting that it is a product of our method as opposed to an inherent characteristic. This creates a peculiar paradox, where users rate the effectiveness based on the method's ability to discover known and useful patterns, but value novel insights for their perceived validity and utility as long as they are easy to understand.

Participants were largely divided about the pattern browser (Figure 4-bottom), with 31% to 46% seeing the tool as beneficial and user friendly against 38% to 46% thinking otherwise. This large range is again supported by the low agreement between participants ($W = 0.11$). In terms of helpfulness and understandability the number of (very) positive reactions are largely in balance with the (very) negative reactions; the helpfulness scores the most positive with almost half of the participants (46%) deeming the browser beneficial for analysing patterns, while a large portion (38%) of participants is also relatively positive about how the browser conveys the patterns in an way that is understandable. Respondents were more critical of the browser's intuitiveness and pleasantness. While a comparable number of participants assessed the intuitiveness as either positive or negative, there were none who were very positive. Conversely, only few negative respondents were very negative (8%) about the pleasantness of the colour scheme used by the interface, whereas most positive participants were very positive (23%).

The factor loadings that belong to these responses are listed in Table 4, and indicate a strong divide between pleasantness ($0.99\lambda_2$) and the other three criteria: intuitiveness ($0.61\lambda_1$), helpfulness ($1.00\lambda_1$), and understandability ($0.85\lambda_1$). A likely explanation is that pleasantness measures purely the visual appearance of the browser, whereas the remaining three are a measure of the browser's usefulness. Intuitiveness stands out, however, by also providing a moderate contribution ($0.29\lambda_2$) to the visual component. This might be explained by that this criterion, like novelty, is a product of the other dimensions rather than an inherent characteristic, suggesting that intuitiveness stems from whether users deem the browser intelligible and easy to use.

Table 2 lists the overall utility of the graph patterns and browser as perceived by the participants, and suggests an overall critical opinion with 15% of the experts agreeing that the patterns and/or browser can be useful. Different from the other responses, this opinion enjoys a much higher, albeit still moderate, agreement ($W = 0.54$). Correlation tests with the participants' familiarity scores show a substantial positive correlation ($\tau = 0.77$) between having a strong negative opinion and having little experience with database terminology, and moderate positive correlation with the unfamiliarity with SPARQL ($\tau = 0.51$) and knowledge graphs ($\tau = 0.42$). This suggests that scholars who

Table 3

Factor loadings of the responses on the presented graph patterns, averaged over participants, using an oblique rotation (*BentlerQ*[5]) with two components which, together, account for 87% of the total variation.

Criterion	λ_1	λ_2
Novelty	-0.54	0.40
Validity	0.88	-0.05
Utility	0.90	0.14
Interpretability	0.03	0.97

Table 4

Factor loadings of the responses on the pattern browser, using an oblique rotation (*Simplimax*[25]) with two components which, together, account for 86% of the total variation.

Criterion	λ_1	λ_2
Intuitiveness	0.61	0.29
Pleasantness	0.02	0.99
Helpfulness	1.00	-0.16
Understandability	0.85	-0.02

Table 5

Confidence of the participants ($W = 0.85$) in relative numbers.

Score	Portion
fully agree	0.00
agree	0.00
neutral	0.38
disagree	0.15
fully disagree	0.46

possess a more inductive, data-focussed, mindset were more positive about our approach, whereas more deductive, theory-minded, scholars were most critical.

Remarks left by the experts shed some light on the results. While a variety of reasons were given, the large majority of these can be summarized as "missing the context". According to these experts, it is difficult to infer anything useful from the patterns if presented in isolation. Rather, more detailed information should be provided on the data and the domain they cover. Other insight that can be gained from the remarks is the strong preference for a natural language representation, rather than the SPARQL format or graph visualization, despite the likely loss of precision due to the translation. A final common remark is the degree of interestingness, which still varies too much.

7. Conclusion & future work

This work introduced an *anytime*, bottom-up, and easily parallelizable algorithm to efficiently discover *generalized multimodal graph patterns* in knowledge graphs. To facilitate further filtering and analysis, the discovered patterns are converted to SPARQL queries and presented in a simple facet browser. An evaluation of the patterns and the browser was held in the form of a user study amongst a select group of domain expert. While reactions were mixed, further analysis suggested that the most critical experts acted from a feeling of uncertainty caused by their unfamiliarity with the technical skills required to fully comprehend the patterns and the method that generated them. Rather, this group expressed their preference for more context and natural language explanations, finding it challenging to interpret the patterns otherwise. Conversely, the experts who did posses appropriate technical backgrounds were more positive in general, particularly where utility is concerned.

Further analysis also revealed a peculiar, yet interesting, paradox that suggests that many experts set out to find interesting new patterns, yet rated novel patterns more negatively because they do not conform to the current scholarly literature or the experts' own beliefs. This effect might be a form of confirmation bias or simply a distrust of new technologies, yet poses an intriguing conundrum since the most straightforward solution (emphasizing existing knowledge) would invalidate the method's entire reason for being. On the other hand, since the perception of novelty appears to emerge from other characteristics rather than being an intrinsic characteristic of a pattern itself, as suggested by our findings, it can be argued that the primary goal should not be about finding *novel* patterns, but rather about discovering explainable connections between patterns that are already known and validated. Developing such methods would be an interesting exercise for future work.

There are several other natural directions to follow up on in future work. First and foremost is the improvement of the measure of interestingness, and how to steer away from uninteresting patterns. This is a common and difficult problem with pattern mining which is largely the result of an algorithm's reliance on statistics. A related problem is the ability to discover *common-sense knowledge*, which, in the context of our method, are patterns of low to moderate frequency that might still convey interesting characteristics. Expanding the method's ability to exploit background information might help address these limitations by making more informed decisions when exploring the search space, for example by favouring patterns that contain elements from a domain-specific taxonomy. Another possible solution to avoid uninteresting patterns might be to more actively involve the users in the discovery process, by asking them to score candidate patterns as they are discovered. This would enable scholars to fine-tuning the output to their own expectations, further increasing explainability and transparency. These scholars can be supported by a meta model that learns to differentiate between patterns that are interesting and those which are not, and which, once satisfactory, can be shared with fellow researchers.

Future work might also consider further improving how scholars can inspect and analyse the discovered patterns. In this work, a first step was made towards an interactive facet browser for this purpose. Our study suggests that the user experience of this browser might be improved, for example, by incorporating detailed information about the context on various levels of granularity. This information could include general statistics about the relevant classes and predicates, as well as provide an overview of their semantics, their members, and other closely related elements. To increase interpretability, the patterns themselves can perhaps be offered as natural language explanations, which could be generated automatically by leveraging the annotations in the graph if provided, or by employing a large language model trained on similar data.

Acknowledgements

The authors would like to express their gratitude to Dr. Sytze van Herck, who has provided valuable feedback and insights during various stages of this research.

References

[1] W. Akhtar, Á. Cortés-Calabuig and J. Paredaens, Constraints in RDF, in: *International Workshop on Semantics in Data and Knowledge Bases*, Springer, 2010, pp. 23–39.

[2] H.E. Bal, D.H.J. Epema, C. de Laat, R. van Nieuwpoort, J.W. Romein, F.J. Seinstra, C. Snoek and H.A.G. Wijshoff, A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term, *Computer* **49**(5) (2016), 54–63. doi:10.1109/MC.2016.127.

[3] M. Barati, Q. Bai and Q. Liu, SWARM: An Approach for Mining Semantic Association Rules from Semantic Web Data, in: *PRICAI 2016: Trends in Artificial Intelligence - 14th Pacific Rim International Conference on Artificial Intelligence, Phuket, Thailand, August 22-26, 2016, Proceedings*, R. Booth and M. Zhang, eds, Lecture Notes in Computer Science, Vol. 9810, Springer, 2016, pp. 30–43. doi:10.1007/978-3-319-42911-3_3.

[4] M. Barati, Q. Bai and Q. Liu, Mining semantic association rules from RDF data, *Knowl. Based Syst.* **133** (2017), 183–196. doi:10.1016/J.KNOSYS.2017.07.009. https://doi.org/10.1016/j.knosys.2017.07.009.

[5] P. Bentler, Factor simplicity index and transformations, *Psychometrika* **42**(2) (1977), 277–295.

[6] A. Cakmak and G. Özsoyoglu, Taxonomy-superimposed graph mining, in: *EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*, A. Kemper, P. Valduriez, N. Mouaddib, J. Teubner, M. Bouzeghoub, V. Markl, L. Amsaleg and I. Manolescu, eds, ACM International Conference Proceeding Series, Vol. 261, ACM, 2008, pp. 217–228. doi:10.1145/1353343.1353372.

[7] D. Calvanese, W. Fischl, R. Pichler, E. Sallinger and M. Simkus, Capturing relational schemas and functional dependencies in RDFS, in: *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[8] V. de Boer, J. Wielemaker, J. van Gent, M. Hildebrand, A. Isaac, J. van Ossenbruggen and G. Schreiber, Supporting Linked Data Production for Cultural Heritage Institutes: The Amsterdam Museum Case Study, in: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, E. Simperl, P. Cimiano, A. Polleres, Ó. Corcho and V. Presutti, eds, Lecture Notes in Computer Science, Vol. 7295, Springer, 2012, pp. 733–747. doi:10.1007/978-3-642-30284-8_56.

[9] T. Declerck, J.P. McCrae, M. Hartung, J. Gracia, C. Chiarcos, E. Montiel-Ponsoda, P. Cimiano, A. Revenko, R. Saurí, D. Lee, S. Racioppa, J.A. Nasir, M. Orlikowski, M. Lanau-Coronas, C. Fäth, M. Rico, M.F. Elahi, M. Khvalchik, M. González and K. Cooney, Recent Developments for the Linguistic Linked Open Data Infrastructure, in: *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk and S. Piperidis, eds, European Language Resources Association, 2020, pp. 5660–5667. https://aclanthology.org/2020.lrec-1.695/.

[10] W. Fan and P. Lu, Dependencies for graphs, in: *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, ACM, 2017, pp. 403–416.

[11] W. Fan, Y. Wu and J. Xu, Functional dependencies for graphs, in: *Proceedings of the 2016 International Conference on Management of Data*, ACM, 2016, pp. 1843–1857.

[12] L.A. Galárraga, C. Teflioudi, K. Hose and F.M. Suchanek, AMIE: association rule mining under incomplete evidence in ontological knowledge bases, in: *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, D. Schwabe, V.A.F. Almeida, H. Glaser, R. Baeza-Yates and S.B. Moon, eds, International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 413–422. doi:10.1145/2488388.2488425.

[13] L. Galárraga, C. Teflioudi, K. Hose and F.M. Suchanek, Fast rule mining in ontological knowledge bases with AMIE+, *VLDB J.* **24**(6) (2015), 707–730. doi:10.1007/S00778-015-0394-1. https://doi.org/10.1007/s00778-015-0394-1.

[14] A. Gangemi and V. Presutti, Towards a pattern science for the semantic web, *Semantic Web* **1**(1–2) (2010), 61–68.

[15] L. Geng and H.J. Hamilton, Choosing the Right Lens: Finding What is Interesting in Data Mining, in: *Quality Measures in Data Mining*, F. Guillet and H.J. Hamilton, eds, Studies in Computational Intelligence, Vol. 43, Springer, 2007, pp. 3–24. doi:10.1007/978-3-540-44918-8_1.

[16] I. Gur, S. Yavuz, Y. Su and X. Yan, DialSQL: Dialogue Based Structured Query Generation, in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, eds, Association for Computational Linguistics, 2018, pp. 1339–1349. doi:10.18653/V1/P18-1124. https://aclanthology.org/P18-1124/.

[17] B. Haslhofer, A. Isaac and R. Simon, Knowledge Graphs in the Libraries and Digital Humanities Domain, in: *Encyclopedia of Big Data Technologies*, S. Sakr and A.Y. Zomaya, eds, Springer, 2019. doi:10.1007/978-3-319-63962-8_291-1.

[18] B. He, L. Zou and D. Zhao, Using conditional functional dependency to discover abnormal data in RDF graphs, in: *Proceedings of Semantic Web Information Management on Semantic Web Information Management*, ACM, 2014, pp. 1–7.

[19] J. Hellings, M. Gyssens, J. Paredaens and Y. Wu, Implication and axiomatization of functional and constant constraints, *Annals of Mathematics and Artificial Intelligence* **76**(3–4) (2016), 251–279.

[20] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier, A.N. Ngomo, A. Polleres, S.M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab and A. Zimmermann, *Knowledge Graphs*, Synthesis Lectures on Data, Semantics, and Knowledge, Morgan & Claypool Publishers, 2021. ISBN 978-3-031-00790-3. doi:10.2200/S01125ED1V01Y202109DSK022.

[21] A. Inokuchi, Mining Generalized Substructures from a Set of Labeled Graphs, in: *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, IEEE Computer Society, 2004, pp. 415–418. doi:10.1109/ICDM.2004.10041.

[22] C. Jiang, F. Coenen and M. Zito, A survey of frequent subgraph mining algorithms, *Knowl. Eng. Rev.* **28**(1) (2013), 75–105. doi:10.1017/S0269888912000331.

[23] M.G. Kendall, A new measure of rank correlation, *Biometrika* **30**(1–2) (1938), 81–93.

[24] M.G. Kendall, Rank correlation methods. (1948).

[25] H.A. Kiers, Simplimax: Oblique rotation to an optimal target with simple structure, *Psychometrika* **59** (1994), 567–579.

[26] M. Kuramochi and G. Karypis, Frequent Subgraph Discovery, in: *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, N. Cercone, T.Y. Lin and X. Wu, eds, IEEE Computer Society, 2001, pp. 313–320. doi:10.1109/ICDM.2001.989534.

[27] J. Lajus, L. Galárraga and F.M. Suchanek, Fast and Exact Rule Mining with AMIE 3, in: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, A. Harth, S. Kirrane, A.N. Ngomo, H. Paulheim, A. Rula, A.L. Gentile, P. Haase and M. Cochez, eds, Lecture Notes in Computer Science, Vol. 12123, Springer, 2020, pp. 36–52. doi:10.1007/978-3-030-49461-2_3.

[28] M. Lissandrini, D. Mottin, K. Hose and T.B. Pedersen, Knowledge graph exploration systems: Are we lost?, in: *CIDR*, Vol. 22, 2022, pp. 10–13.

[29] K. Mandemakers, G. Bloothooft, F. Laan, J. Raad, R.J. Mourits, R.L. Zijdeman et al., LINKS. A System for Historical Family Reconstruction in the Netherlands, *Historical Life Course Studies* **13** (2023), 148–185.

[30] T. Martin, V. Fuentes, P. Valtchev, A.B. Diallo and R. Lacroix, Generalized graph pattern discovery in linked data with data properties and a domain ontology, in: *SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, April 25 - 29, 2022*, J. Hong, M. Bures, J.W. Park and T. Cerný, eds, ACM, 2022, pp. 1890–1899. doi:10.1145/3477314.3507301.

[31] C. Meghini, R. Scopigno, J. Richards, H. Wright, G. Geser, S. Cuy, J. Fihn, B. Fanini, H. Hollander, F. Niccolucci, A. Felicetti, P. Ronzino, F. Nurra, C. Papatheodorou, D. Gavrilis, M. Theodoridou, M. Doerr, D. Tudhope, C. Binding and A. Vlachidis, ARIADNE: A Research Infrastructure for Archaeology, *ACM Journal on Computing and Cultural Heritage* **10**(3) (2017), 18:1–18:27. doi:10.1145/3064527.

[32] C. Meilicke, M.W. Chekol, D. Ruffinelli and H. Stuckenschmidt, An Introduction to AnyBURL, in: *KI 2019: Advances in Artificial Intelligence - 42nd German Conference on AI, Kassel, Germany, September 23-26, 2019, Proceedings*, C. Benzmüller and H. Stuckenschmidt, eds, Lecture Notes in Computer Science, Vol. 11793, Springer, 2019, pp. 244–248. doi:10.1007/978-3-030-30179-8_20.

[33] A. Meroño-Peñuela, V. de Boer, M. van Erp, R. Zijdeman, R. Mourits, W. Melder, A. Rijpma and R. Schalk, CLARIAH: Enabling Interoperability Between Humanities Disciplines with Ontologies, in: *Applications and Practices in Ontology Design, Extraction, and Reasoning*, G. Cota, M. Daquino and G.L. Pozzato, eds, Studies on the Semantic Web, Vol. 49, IOS Press, 2020, pp. 73–90. doi:10.3233/SSW200036.

[34] R.G. Miani, C.A. Yaguinuma, M.T. Santos and M. Biajiz, Narfo algorithm: Mining non-redundant and generalized association rules based on fuzzy ontologies, *Enterprise Information Systems* (2009), 415–426.

[35] S.H. Muggleton, Inductive Logic Programming, *New Gener. Comput.* **8**(4) (1991), 295–318. doi:10.1007/BF03037089.

[36] S.H. Muggleton, Inverse Entailment and Progol, *New Gener. Comput.* **13**(3&4) (1995), 245–286. doi:10.1007/BF03037227.

[37] S.H. Muggleton and C. Feng, Efficient Induction of Logic Programs, in: *Algorithmic Learning Theory, First International Workshop, ALT '90, Tokyo, Japan, October 8-10, 1990, Proceedings*, S. Arikawa, S. Goto, S. Ohsuga and T. Yokomori, eds, Springer/Ohmsha, 1990, pp. 368–381.

[38] S.H. Muggleton, L.D. Raedt, D. Poole, I. Bratko, P.A. Flach, K. Inoue and A. Srinivasan, ILP turns 20 - Biography and future challenges, *Mach. Learn.* **86**(1) (2012), 3–23. doi:10.1007/S10994-011-5259-2. https://doi.org/10.1007/s10994-011-5259-2.

[39] V. Nebot and R.B. Llavori, Finding association rules in semantic web data, *Knowl. Based Syst.* **25**(1) (2012), 51–62. doi:10.1016/J.KNOSYS.2011.05.009. https://doi.org/10.1016/j.knosys.2011.05.009.

[40] A.G. Nuzzolese, A. Gangemi, V. Presutti and P. Ciancarini, Encyclopedic knowledge patterns from wikipedia links, in: *International Semantic Web Conference*, Springer, 2011, pp. 520–536.

[41] A.G. Nuzzolese, V. Presutti, A. Gangemi, S. Peroni and P. Ciancarini, Aemoo: Linked data exploration based on knowledge patterns, *Semantic Web* **8**(1) (2016), 87–112.

[42] R. Palme and P. Welke, Frequent Generalized Subgraph Mining via Graph Edit Distances, in: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2022, Grenoble, France, September 19-23, 2022, Proceedings, Part II*, I. Koprinska, P. Mignone, R. Guidotti, S. Jaroszewicz, H. Fröning, F. Gullo, P.M. Ferreira, D. Roqueiro, G. Ceddia, S. Nowaczyk, J. Gama, R.P. Ribeiro, R. Gavaldà, E. Masciari, Z.W. Ras, E. Ritacco, F. Naretto, A. Theissler, P. Biecek, W. Verbeke, G. Schiele, F. Pernkopf, M. Blott, I. Bordino, I.L. Danesi, G. Ponti, L. Severini, A. Appice, G. Andresini, I. Medeiros, G. Graça, L.A.D. Cooper, N. Ghazaleh, J. Richiardi, D.S. Miranda, K. Sechidis, A. Canakoglu, S. Pidò, P. Pinoli, A. Bifet and S. Pashami, eds, Communications in Computer and Information Science, Vol. 1753, Springer, 2022, pp. 477–483. doi:10.1007/978-3-031-23633-4_32.

[43] A. Petermann, G. Micale, G. Bergami, A. Pulvirenti and E. Rahm, Mining and ranking of generalized multi-dimensional frequent subgraphs, in: *Twelfth International Conference on Digital Information Management, ICDIM 2017, Fukuoka, Japan, September 12-14, 2017*, IEEE, 2017, pp. 236–245. doi:10.1109/ICDIM.2017.8244685.

[44] A. Polleres, From SPARQL to rules (and back), in: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, C.L. Williamson, M.E. Zurko, P.F. Patel-Schneider and P.J. Shenoy, eds, ACM, 2007, pp. 787–796. doi:10.1145/1242572.1242679.

[45] J.R. Quinlan, Learning Logical Definitions from Relations, *Mach. Learn.* **5** (1990), 239–266. doi:10.1007/BF00117105.

[46] R. Ramezani, M. Saraee and M. Nematbakhsh, SWApriori: a new approach to mining Association Rules from Semantic Web Data, *Journal of Computing and Security* **1** (2014), 16.

[47] S. Schenk, A SPARQL Semantics Based on Datalog, in: *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007, Proceedings*, J. Hertzberg, M. Beetz and R. Englert, eds, Lecture Notes in Computer Science, Vol. 4667, Springer, 2007, pp. 160–174. doi:10.1007/978-3-540-74565-5_14.

[48] C. Schöch, Big? Smart? Clean? Messy? Data in the Humanities?, *Journal of the Digital Humanities* **2**(3) (2013).

[49] F. Shen, H. Liu, S. Sohn, D.W. Larson and Y. Lee, BmQGen: Biomedical query generator for knowledge discovery, in: *2015 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2015, Washington, DC, USA, November 9-12, 2015*, J. Huan, S. Miyano, A. Shehu, X.T. Hu, B. Ma, S. Rajasekaran, V.K. Gombar, M. Schapranow, I. Yoo, J. Zhou, B. Chen, V. Pai and B.G. Pierce, eds, IEEE Computer Society, 2015, pp. 1092–1097. doi:10.1109/BIBM.2015.7359833.

[50] P.A. Szekely, C.A. Knoblock, F. Yang, X. Zhu, E.E. Fink, R. Allen and G. Goodlander, Connecting the Smithsonian American Art Museum to the Linked Data Cloud, in: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, P. Cimiano, Ó. Corcho, V. Presutti, L. Hollink and S. Rudolph, eds, Lecture Notes in Computer Science, Vol. 7882, Springer, 2013, pp. 593–607. doi:10.1007/978-3-642-38288-8_40.

[51] S. Wang, H. Scells, B. Koopman and G. Zuccon, Can ChatGPT Write a Good Boolean Query for Systematic Review Literature Search?, in: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, H. Chen, W.E. Duh, H. Huang, M.P. Kato, J. Mothe and B. Poblete, eds, ACM, 2023, pp. 1426–1436. doi:10.1145/3539618.3591703.

[52] X. Wilcke, V. de Boer, M.T.M. de Kleijn, F. van Harmelen and H.J. Scholten, User-centric pattern mining on knowledge graphs: An archaeological case study, *J. Web Semant.* **59** (2019). doi:10.1016/J.WEBSEM.2018.12.004. https://doi.org/10.1016/j.websem.2018.12.004.

[53] X. Wilcke, M. de Kleijn, V. de Boer, H.J. Scholten and F. van Harmelen, Bottom-up Discovery of Context-aware Quality Constraints for Heterogeneous Knowledge Graphs, in: *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2020, Volume 1: KDIR, Budapest, Hungary, November 2-4, 2020*, A.L.N. Fred and J. Filipe, eds, SCITEPRESS, 2020, pp. 81–92. doi:10.5220/0010113500810092.

[54] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* **3**(1) (2016), 1–9.

[55] Y. Yu and J. Heflin, Extending functional dependency to detect abnormal data in RDF graphs, in: *International Semantic Web Conference*, Springer, 2011, pp. 794–809.

[56] Z. Zhang and O. Nasraoui, Mining search engine query logs for query recommendation, in: *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, L. Carr, D.D. Roure, A. Iyengar, C.A. Goble and M. Dahlin, eds, ACM, 2006, pp. 1039–1040. doi:10.1145/1135777.1136004.