information

MDPI

Article

Extraction Patterns to Derive Social Networks from Linked Open Data using SPARQL †

Raji Ghawi * and Jürgen Pfeffer

Bavarian School of Public Policy, Technical University of Munich, 80333 Munich, Germany; juergen.pfeffer@tum.de

* Correspondence: raji.ghawi@tum.de

† This paper is an extended version of our paper published in the proceedings of the 24th International Conferences on Conceptual Structures, Marburg, Germany, 1–4 July 2019.

Received: 7 June 2020; Accepted: 2 July 2020; Published: 12 July 2020

Abstract: Linked Open Data (LOD) refers to freely available data on the World Wide Web that are typically represented using the Resource Description Framework (RDF) and standards built on it. LOD is an invaluable resource of information due to its richness and openness, which create new opportunities for many areas of application. In this paper, we address the exploitation of LOD by utilizing SPARQL queries in order to extract social networks among entities. This enables the application of de-facto techniques from Social Network Analysis (SNA) to study social relations and interactions among entities, providing deep insights into their latent social structure.

Keywords: linked open data; social networks; RDF; SPARQL algebra; extraction patterns

1. Introduction

This paper is an extension of an already published conference paper [1], where we investigated how to extract social networks from Linked Open Data. In recent years, the Web has evolved from a network of linked documents to one where both documents and data are linked, resulting in what is commonly known as the Web of Data. Underpinning this evolution is a set of best practices known as Linked Open Data (LOD) [2], which provide mechanisms for publishing and connecting structured data on the Web in a machine-readable form with explicit semantics. Recently, Linked Open Data has evolved from an academic endeavor into one that has been embraced by numerous governments and industrial stakeholders.

Due to the creation of an increasing number of publicly available Linked Open Data resources, the Web of Data has become a major application area for semantic technologies. Currently, the so-called LOD cloud contains over 1200 datasets, with billions of facts from many different domains like geography, media, biology, chemistry, economy, energy, etc., and millions of links among entities (http://lod-cloud.net). Examples of large LOD datasets are DBpedia [3,4] (3.4 million entities, 1 billion facts), and YAGO [5] (17 million entities, 150 million facts).

All such data are typically represented using the Resource Description Framework (RDF) which is the World Wide Web Consortium's (W3C) standard language for representing information in the Semantic Web [6,7]. RDF is based on a directed graph data model, where both nodes and edges are labeled. An RDF graph is a set of triples of the form (s, p, o), which can be interpreted as edges labeled by p (the predicate) from nodes labeled by s (the subject) to nodes labeled by o (the object). The elements of a triple are typically Internationalized Resource Identifiers (IRIs)-global names that uniquely identify resources on the Web. SPARQL, which became a W3C recommendation in 2008, is the standard query language for RDF [8,9]. Recently, a new version of the SPARQL query language,

called SPARQL 1.1, has been standardized by W3C. It addresses some of the limitations of the original language by introducing a wide range of constructs [10].

The increasing adoption of Linked Open Data is turning the Web into a global data space that connects data from diverse domains and enables genuinely novel applications. The richness and openness of Linked Open Data make it an invaluable resource of information, and creates new opportunities for many areas of application. For instance, in this present work, we address the exploitation of Linked Open Data in order to extract social networks among entities. This will enable the application of de-facto techniques from Social Network Analysis to study social relations and interactions among entities, providing deep insights into their latent social structure.

Social Network Analysis (SNA) refers to the collection of methods, techniques, and tools in sociometry aiming at the analysis of social networks. There is an abundance of tools allowing for the analysis and visualization of such networks. A social network may be dense or not, the "social distances" among individuals may be short or long, etc. An individual may be "central" (directly linked to many other individuals) or an "isolate" (not linked to others). However, more subtle notions are also possible, e.g., an individual who is only linked to people having many relationships is considered to be a more powerful node in the network than an individual having many connections to less connected individuals.

The work presented in this paper is an attempt to bring together the two research areas of Linked Open Data and Social Network Analysis. The main idea is to derive social networks from large datasets of linked open data, such that extracted networks become a fresh material for study and analysis, while at the same time forming an additional asset of knowledge added to linked open data.

The main contributions of this present paper are the following:

- We propose several techniques to extract social networks from linked open data.
- We express those techniques in a formal way using SPARQL algebra.
- We present formal translations into social networks.
- We present several case studies that apply some of the presented techniques.

The paper is organized as follows. Section 2 presents a motivation example that demonstrates the importance of the proposed approach, while Section 3 gives an overview of relevant previous work. Section 4 provides a background on social networks and SPARQL algebra. Sections 5 and 6 are the core of this paper, introducing network extraction patterns for complete networks as SPARQL queries. Then, a generic translation method that transforms query results into networks is presented in Section 7. Section 8 concludes the paper with an overall discussion.

2. Motivation

To motivate our work, we consider the relation between actors and movies they acted in. In the YAGO dataset, this relation is expressed using the `yago:actedIn` predicate, which relates an actor (subject) to a movie (object) he/she acted in. Table 1 shows a subset of RDF triples from the YAGO dataset about this relation. Thus, we can visually represent this relation as a two-mode affiliation network (two types of nodes). Figure 1 demonstrates a smaple of such a network obtained from triples in Table 1.

More interestingly, indirect relations can be derived from this two-mode affiliation relation. That is, two-mode networks are often transformed into one-mode networks (only one type of nodes) using a procedure which is often referred to as *projection* [11,12]. Projection is done by selecting one of the sets of nodes and linking two nodes from that set if they are connected to the same node (of the other kind). In the case of actors and movies, we can derive an actor–actor relationship if they act in the same movie. For example, Jodie Foster and Matthew McConaughey would be connected as they have acted in the movie *Contact*. Traditionally, the ties in projected one-mode networks do not have weights attached to them. However, recent empirical studies of two-mode networks have created a weighted one-mode network by defining the weights as the number of co-occurrences (e.g., the number of

movies in which two actors have co-acted). Figure 2 shows the co-acting network derived from the actor-movie two-mode network. For instance, the connection between Penolope Cruz and Steve Zahn has a weight of 2 as they acted in two movies: *Sahara* and *Bandidas*.

Table 1. A subset of RDF triples from YAGO with `yago:actedIn` predicate.

Jodie_Foster	actedIn	Contact_(1997_US_film)
Matthew_McConaughey	actedIn	Contact_(1997_US_film)
Matthew_McConaughey	actedIn	Sahara_(2005_film)
Penélope_Cruz	actedIn	Sahara_(2005_film)
Steve_Zahn	actedIn	Sahara_(2005_film)
Penélope_Cruz	actedIn	Bandidas
Salma_Hayek	actedIn	Bandidas
Steve_Zahn	actedIn	Bandidas
...	...	...

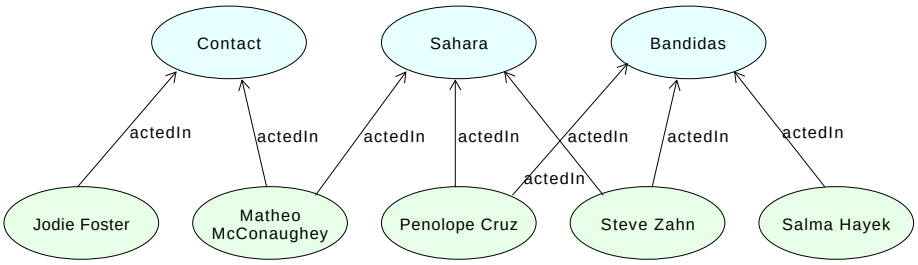

Figure 1. Sample RDF Graph of `yago:actedIn` predicate in YAGO.

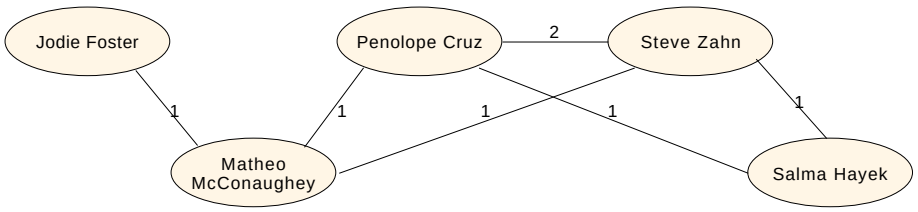

Figure 2. Extracted social network of co-acting among actors.

The purpose of this present work is to investigate possible ways to extract such social networks from linked open data (expressed in RDF), and to present such ways as systematic techniques using: (1) SPARQL queries formally expressed in SPARQL algebra, and (2) formal transformations of the query results into networks.

3. Related Work

The richness and openness of Linked Open Data, as well as the inter-linking of the many datasets, known as LOD cloud, make it an invaluable resource of information, and create new opportunities for many areas of application. This leads to an increasing adoption of LOD by the scientific community, and several sectors of industry [13]. Among others, one of the major factors that foster the evolution and adoption of LOD is the semantic technologies (RDF [7], OWL [14], and SPARQL [10]) standardized by W3C. Being structured using a standard data format (RDF), the consumption of Linked Open Data is facilitated with SPARQL, a standard query language, and protocol to access RDF datasets. SPARQL is based on a solid background with respect to its syntax and semantics [9] (see Section 4.2 below). The large amount of RDF data available on the Web is exposed by means of (a) Linked Data-enabled dereferenceable URIs in various formats (such as RDF/XML, Turtle, RDFa, etc.) and by (b) SPARQL endpoints (SPARQL endpoints are RESTful web services that accept SPARQL queries over HTTP

adhering to the SPARQL protocol, as defined by the respective W3C recommendations [15]). Most of the LOD datasets are interlinked, which allows navigating through them and facilitates building complex queries by combining data from different, sometimes heterogeneous and often physically distributed datasets. To address this use case, the W3C recommendation defines a federation extension [16] for SPARQL 1.1 [10], which allows for combining graph patterns that can be evaluated over several endpoints within a single query [15].

Several areas of application are increasingly benefiting from the large amount of RDF data available in the Web of Data, and exploiting their potential power. For instance, Recommender Systems are among such applications consuming Linked Open Data. Passant [17] proposes a Music recommender system, called *drec*, which is built on top of DBpedia. Di Noia et al. [18] develop a content-based recommender system that leverages the data available within Linked Open Data datasets in order to recommend movies to the end users.

Some works seek to combine social analytics with the Linked Open Data (LOD) cloud. De Vocht et al. [19] propose a semantically driven aggregation of social data, where they use semantic technologies, common vocabularies, and Linked Open Data to extract and mine the data about scientific events out of context of microblogs (e.g., Twitter). As a proof-of-concept, they implement and evaluate a researcher profiling use case. Razis et al. [20,21] propose an ontology schema towards linking semantified Twitter social analytics with the Linked Open Data cloud. The ontology is deployed over a publicly available service that measures how influential a Twitter account is by combining its social activity in Twitter. They also introduce in [22] a methodology for discovering and suggesting similar Twitter accounts, based entirely on their disseminated content in terms of used Twitter entities (mentions, replies, hashtags, URLs). The methodology is based on semantic representation protocols and related technologies. An ontological schema is also described towards the semantification of the Twitter accounts and their entities.

Several works in the literature have already attempted to combine Social Network Analysis with semantic technologies. For instance, Flink [23] is an early system for the extraction, aggregation, and visualization of online social networks. Flink employs semantic technology for reasoning with social information aggregated from disparate sources: web pages, emails, publication archives, and FOAF profiles. Martin et al. [24] propose a model to represent social networks in RDF and show how SPARQL can be used to query and transform networks. However, the proposed data model is unnecessarily complex as relations among nodes are represented as RDF resources, hence additional predicates are introduced to link nodes to the relations. Moreover, at that time, aggregation was missing in SPARQL, therefore SQL is used in the model. Other works have been proposed to use SPARQL and other semantic technologies not only to represent social networks, but also to perform social network analysis [25–27]. However, all the aforementioned works use semantic technologies to represent social networks and/or to perform social network analysis. Unlike our work, none of them extract social networks from RDF datasets (LOD). Our work focuses on network extraction patterns from RDF, not on representing the networks themselves.

Groth and Gil [28] present an approach for extracting networks from Linked Data, where extracted networks can then be analyzed through network analysis algorithms, and the results of these analyses can be published back as Linked Data. Zehetner [29] proposes in his dissertation a framework, called *SocioCatcher*, to extract and analyze social networks from DBpedia. However, both of these works focus on the system and its computational workflows, without a solid theoretical basis and formalism of extraction patterns as we do in our present work.

4. Background

4.1. Social Networks

"A social network consists of a finite set or sets of actors and relation or relations defined on them." ([30]). Social networks can be classified based on the set of actors and their environment

into: (1) complete networks, and (2) ego centered networks. A complete network addresses an entire population, where the individuals define each other's environment [11]. Ego-centric networks address an identified individual (ego) and his environment. When we deal with social networks extracted from LOD, we also distinguish between complete and partial networks [11]. Complete networks cover an entire population of individuals in the entire dataset, whereas a partial network would cover a subset of the population defined by means of a specific context, e.g., time, location, or gender, etc. For example, let us consider a co-acting relationship among actors who acted in same movies. When we extract all co-acting relations among all actors (as defined in the dataset), we obtain a *complete network*, while when we extract such relations for Indian actors only, or for movies produced in the 1990s, we obtain a *partial network*. Moreover, when we extract such relations for Jodie Foster and her co-actors, we obtain an *ego-centric* network [11].

Typically, networks are represented in terms of *graphs*. A graph G is a pair (V, E) that consists of a set V of vertices, and a set E of edges. While the elements of V represent the actors of the network, the ties among them are represented in E. Therefore, an edge is simply a pair of vertices $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$. If the relation between a pair of vertices $i, j \in V$ is asymmetric, we say the edges are *directed* (and so is the network). Otherwise, the edges are bidirectional and the network is said to be *undirected*. Relations among vertices could have a sort of strength, in this case, edges are given numeric *weights*, and we say the network is *weighted*. A weighted network is represented as a triple $G = (V, E, \omega)$ where: ω is a function $\omega : E \to \mathbb{R}$ that maps edges to their weight values.

In Social Network Analysis literature, many metrics (indices) have been developed to characterize social networks, at both (a) *node level*, such as: (in-, out-) degree, and centrality (closeness, betweenness), and (b) *network level*, such as: density, diameter, average degree, average path length, and average clustering coefficient [30]. Moreover, advanced analysis can also be applied onto social networks, including for example: community detection, diffusion dynamics, and link prediction.

4.2. SPARQL Algebra

Let **I**, **L**, and **B** be pairwise disjoint sets of IRIs, literals, and blank nodes, respectively, where literals can be numbers, strings, or Boolean values. The set **T** of (RDF) terms is $\mathbf{I} \cup \mathbf{L} \cup \mathbf{B}$. An RDF triple is an element (s, p, o) of $(\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times \mathbf{T}$, with s called the subject, p the predicate, and o the object. An RDF graph is a finite set of RDF triples.

We adopt the SPARQL algebra from Kaminski et al. [31,32], which is based on the SPARQL 1.1 specification [10], and presents a formalisation that makes ambiguous aspects of the specification precise. We distinguish three types of building blocks: expressions, patterns, and queries that are built over terms **T** and an infinite set $\mathbf{X} = \{?x, ?y, \cdots\}$ of variables, disjoint from **T**.

Expressions in SPARQL are inductively defined as follows:

- all variables in **X** and all terms in $\mathbf{I} \cup \mathbf{L}$ are expressions;
- if $?x \in \mathbf{X}$, then $\texttt{bound}(?x)$ is an expression;
- if $t \in \mathbf{T} \cup \mathbf{X}$, then $\texttt{isIRI}(t)$, $\texttt{isLiteral}(t)$, and $\texttt{isBlank}(t)$ are expressions;
- if E_1 and E_2 are expressions, then so are: $(E_1 + E_2)$, $(E_1 - E_2)$, $(E_1 * E_2)$, (E_1 / E_2), $(E_1 \doteq E_2)$, $(E_1 < E_2)$, $(E_1 > E_2)$, $(\neg E)$, $(E_1 \wedge E_2)$, and $(E_1 \vee E_2)$;
- $\texttt{exists}(P)$ is an expression, if P is a pattern.

Patterns in SPARQL are inductively defined as follows:

- a *basic graph pattern (BGP)* is a set of triple patterns, that is, elements of the set

$$(\mathbf{I} \cup \mathbf{L} \cup \mathbf{X}) \times (\mathbf{I} \cup \mathbf{X}) \times (\mathbf{I} \cup \mathbf{L} \cup \mathbf{X})$$

- $Join(P_1, P_2)$, $Union(P_1, P_2)$ and $SetMinus(P_1, P_2)$ are patterns if P_1 and P_2 are patterns;
- $Filter(E, P)$ is a pattern if P is a pattern and E is an expression;
- $LeftJoin(E, P_1, P_2)$ is a pattern if P_1 and P_2 are patterns and E is an expression;

- $GroupAgg(Z,?x, f, E, P)$, where Z is a set of variables, called grouping variables, $?x$ is a variable called aggregation variable, f is an aggregate function, E is an expression, and P is a pattern;
- $Extend(?x, E, P)$ is a pattern (which captures `BIND` and `VALUES` constructs), where $?x$ is a variable, E is an expression, and P is a pattern;

The construct $GroupAgg$ is close to the grouping operator in the relational algebra, where Z represents the set of grouping variables, $?x$ is the fresh variable storing the aggregation result, f is the aggregate function (such as: `Count`, `Sum`, `Avg`, `Min`, or `Max`), and E is the expression (often a variable) we are aggregating over.

Queries in SPARQL are expressions of the form $Project(X, P)$ or $Distinct(Project(X, P))$, for P a pattern and X a set of variables (called free variables).

The semantics of SPARQL is defined in terms of (solution) mappings that is, partial functions μ from variables **X** to terms **T**. The domain of μ, denoted $dom(\mu)$, is the set of variables over which μ is defined. The solution of a SPARQL query Q over an RDF graph G is a multiset of mappings $\mathcal{M} = [\![P]\!]_G$, where $[\![.]\!]_G$ is an evaluation function that maps queries and RDF graphs to multisets of solution mappings.

5. Network Extraction Techniques

In this section and the next one, we introduce techniques for extracting complete social networks from LOD using SPARQL. Those techniques are basically based on identification of common RDF triple patterns and how the network ties are inferred from such patterns.

We classify those techniques based on the number of predicates and number of RDF triples in a pattern. We start in this section with patterns having one predicate p, where we present possible patterns composed of one and two RDF triples, all having p as predicate (Figure 3). Then, in the next section, we discuss patterns having two predicates p, q, where we present possible patterns for two, three, and four RDF triples having p, q predicates.

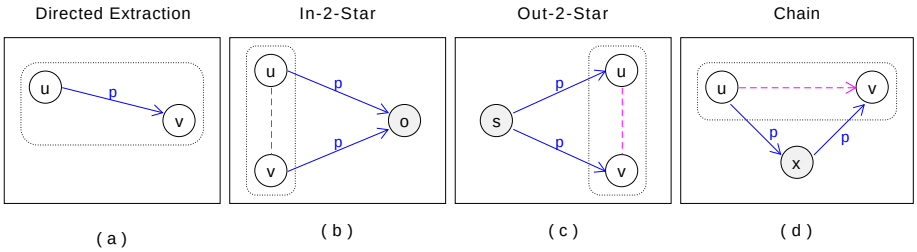

Figure 3. Extraction patterns with one predicate. (**a**) Direct Extraction: one direct connection from u to v. (**b**) In-2-Star: two connections are coming-into o from u and v. (**c**) Out-2-Star: two connections are going-out s to u and v. (**d**) Chain: two successive connections between u and v via x.

5.1. Extraction Using One Predicate

We start with patterns having one predicate $p \in \mathbf{I}$. We present the possible patterns composed of one, and two RDF triples, having p as predicate.

5.2. Pattern with One Triple (Direct Extraction)

Given certain predicate $p \in \mathbf{I}$, let $?u, ?v \in \mathbf{X}$ be two variables, then the BGP $(?u, p, ?v)$ represents a direct relation (tie) from a node represented by $?u$ to a node represented by $?v$ using the predicate p, as shown in Figure 3a. We call this case: *Direct Extraction*. Thus, to extract all such ties among nodes in the RDF graph using p, the following SPARQL query can be used:

$$Q_1 = Project(\{?u, ?v\}, \{(?u, p, ?v)\}) \quad (1)$$

which can be translated into SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?u p ?v. }
```

The extracted relation in this case is always *directional*, i.e., the order of nodes does matter (Unless the predicate p is *symmetric*, such as `yago:isMarriedTo`, then the relation becomes bidirectional). This means the extracted social network will be a directed network. In this type of extraction, one tie in the network is extracted from one triple in the RDF graph.

Example 1. *The YAGO dataset has a predicate* `yago:influences` *that relates persons to others who have influence on them (on their opinions or work). To extract a social network based on this* influence *relationship, we can use this query:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{yago:influences}, ?v)\}$$

The extracted network has been analyzed in [33].

5.3. Patterns with Two Triples

Social networks can be indirectly extracted from RDF data when the relation is *implicit* in the RDF graph. A tie between two nodes can be derived from several triples, as we saw in the motivation example (Section 2). When we derive a tie between two nodes using two RDF triples, it does matter where the positions of those nodes in the RDF triples are: subject–subject, object–object, subject–object, or object–subject. Therefore, we distinguish three derivation patterns (variants), namely: in-2-star, out-2-star, and chain.

5.3.1. In-2-Star

A tie between two nodes u, v is derived when both nodes are linked to a third node o, that is, when u and v are in the *subject* position of two RDF triples having the same *object* o and the same predicate p (Figure 3b). This pattern $(u, p, o), (v, p, o)$ is named in-2-star because it resembles a *star* network whose center is o and has 2 peripheries u, v directed *in*-to the center: $u \rightarrow o \leftarrow v$.

The following SPARQL query can be used to extract all such ties among nodes in the RDF graph using a predicate $p \in \mathbf{I}$, and three variables $?u, ?v, ?o \in \mathbf{X}$:

$$Q_2 = Project(\{?u, ?v\}, \{(?u, p, ?o), (?v, p, ?o)\}), \tag{2}$$

which can be translated into SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?u p ?o. ?v p ?o.}
```

In this case, the derived relation is bidirectional; therefore, in contrast to direct extraction, derivation using the same predicate with in-2-star pattern always yields *undirected* networks.

Example 2. *The DBpedia dataset has a predicate* `dbo:almaMater` *that relates people to academic institutions they studied in. This predicate can be used to derive a so-called* alumni *relationship among people who attended the same university. The following query can be used for this purpose:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{dbo:almaMater}, ?o), (?v, \texttt{dbo:almaMater}, ?o)\})$$

Example 3. *Our motivation example in Section 2 belongs to this type of derivation. YAGO predicate* `yago:actedIn` *can be used to derive co-acting network among actors who acted in same movies. The query can be expressed as:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{yago:actedIn}, ?o), (?v, \texttt{yago:actedIn}, ?o)\})$$

Networks derived by the same predicate can be weighted when the number of co-occurrences of the in-2-star pattern is taken into consideration. For instance, in co-acting network, the weight of a tie between two actors is the number of movies they commonly acted in. In order to express this relation in a SPARQL query, we need a group-aggregation algebraic pattern. Kaminski's SPARQL algebra provides $GroupAgg(Z, ?x, f, E, P)$ pattern for this purpose, where Z is the set of grouping variables, $?x$ is aggregation variable, f is aggregation function, E is aggregation expression, and P is a pattern. Thus, the query needed to extract a weighted network using derivation by the same predicate with in-2-star pattern can be written as:

$$Q_3 = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?o\rangle, \{(?u, p, ?o), (?v, p, ?o)\})) \quad (3)$$

In this query, the list of grouping variables is $Z = \langle ?u, ?v\rangle$ (as we want to count the number of their co-occurrences), the aggregation variable is $?w$, a fresh variable to store the weight of the tie, the aggregation function f is `Count`, the aggregation expression is simply the variable $?o$, and the pattern P is $\{(?u, p, ?o), (?v, p, ?o)\}$. Finally, we select the three variables u, v and w. The query can be syntactically expressed as:

```
SELECT ?u ?v (COUNT(?o) AS ?w) WHERE { ?u p ?o. ?v p ?o. } GROUP BY ?u ?v
```

5.3.2. Out-2-Star

With out-2-star pattern, a tie between two nodes u, v is derived when both nodes have incoming links from a third node s, that is, when u and v are in the *object* position of two RDF triples having the same *subject* s and the same predicate p (Figure 3c). This pattern $(s, p, u), (s, p, v)$ is named out-2-star because it resembles a *star* network whose center is s and has 2 peripheries u, v directed *out* from the center: $u \leftarrow s \rightarrow v$.

The following SPARQL query can be used to extract all such ties among nodes in the RDF graph using a predicate $p \in \mathbf{I}$, and three variables $?u, ?v, ?s \in \mathbf{X}$:

$$Q_4 = Project(\{?u, ?v\}, \{(?s, p, ?u), (?s, p, ?v)\}) \quad (4)$$

In this case, the derived relation is also bidirectional, and hence the extracted network is *undirected*. This query can be translated into SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?s p ?u. ?s p ?v.}
```

The BGP can be shortened further into `{ ?s p ?u, ?v.}` since the two triple patterns share the same subject and predicate.

Example 4. *The DBpedia dataset has a predicate* `dbo:starring` *that relates a movie (or TV show) to people who have a starring role in it. This predicate can be used to derive a relationship among actors who have starring roles in same movies. The following query can be used for this purpose:*

$$Q = Project(\{?u, ?v\}, \{(?s, \texttt{dbo:starring}, ?u), (?s, \texttt{dbo:starring}, ?v)\})$$

Weighted networks can also be extracted using this pattern by taking into consideration the number of co-occurrences. The corresponding SPARQL query is as follows:

$$Q_5 = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?s\rangle, \{(?s, p, ?u), (?s, p, ?v)\})) \quad (5)$$

Note that the aggregation expression here is $\langle ?s\rangle$. Syntactically, the query can be expressed as:

```
SELECT ?u ?v (COUNT(?s) AS ?w) WHERE { ?s p ?u, ?v. } GROUP BY ?u ?v
```

5.3.3. Chain

In the *chain* pattern, a tie between two nodes u, v is derived when there is a chain of two links starting from one of these nodes, ending at the other, and traversing a third node x. This case for instance occurs when u is the *subject* of a triple, v is the *object* of another triple, and x is the *object* of the first triple and the *subject* of the other, and, of course, p is the predicate in both triples. This pattern $(u, p, x), (x, p, v)$ is named chain because it resembles to a *chain* starting at u, traversing x, and ending at v: $u \rightarrow x \rightarrow v$. The SPARQL query used to extract all such ties using a predicate $p \in \mathbf{I}$, and three variables $?u, ?v, ?x \in \mathbf{X}$ is:

$$Q_6 = Project(\{?u, ?v\}, \{(?u, p, ?x), (?x, p, ?v)\}) \tag{6}$$

Syntactically, the query can be expressed as:

```
SELECT ?u ?v WHERE { ?u p ?x. ?x p ?v.}
```

In this case, the derived relation is directed according to the original direction of links in the RDF triples. For example, the extracted tie is directed from u to v because u is the subject of the first triple and v is the object of the other one. Therefore, similar to direct extraction, derivation using the same predicate with chain pattern always yields *directed* networks. The two triples $\{(?u, p, ?x), (?x, p, ?v)\}$ can be merged using a property sequence path [10], yielding the triple pattern $(?u, p/p, ?v)$ as a result. This is possible because the variable $?x$ serves as the object of the first triple, and as the subject of the second. Hence, the query can be expressed as:

```
SELECT ?u ?v WHERE { ?u p/p ?v.}
```

Example 5. *The DBpedia dataset has a predicate* `dbo:parent` *that relates a person to his parent(s). This predicate can be used to derive a* grandparent *relationship among people. A person u has a grandparent v, when the parent of u is x and the parent of x is v. The corresponding query is:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{dbo:parent}, ?x), (?x, \texttt{dbo:parent}, ?v)\})$$

Sometimes, depending on the predicate p, the extracted network using chain pattern could be weighted. The corresponding SPARQL query is as follows (with the aggregation expression being $\langle ?x \rangle$):

$$Q_7 = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v \rangle, ?w, Count, \langle ?x \rangle, \{(?u, p, ?x), (?x, p, ?v)\})) \tag{7}$$

Syntactically, this query can be expressed as:

```
SELECT ?u ?v (COUNT(?x) AS ?w) WHERE { ?u p ?x. ?x p ?v } GROUP BY ?u ?v
```

6. Extraction Using Two Predicates

In this section, we continue with patterns having two predicates $p, q \in \mathbf{I}$. We present the possible patterns composed of two, three, and four RDF triples, having p and q predicates.

6.1. Patterns with Two Triples

In this type of derivation, a tie is extracted from two RDF triples having two different predicates $p, q \in \mathbf{I}$. Similarly to derivation using same predicate, we distinguish three patterns according to the positions of nodes in the triples, namely: in-2-star, out-2-star, and chain.

6.1.1. In-2-Star

This pattern is similar to the in-2-star pattern in derivation using same predicate: a tie between two nodes u, v is derived when both nodes are linked to a third node o, that is, when u and v are in the *subject* position of two RDF triples having the same *object* o (Figure 4a). However, the difference here is that the two triples have different predicates p and q, thus the basic graph pattern is: $\{(u, p, o), (v, q, o)\}$.

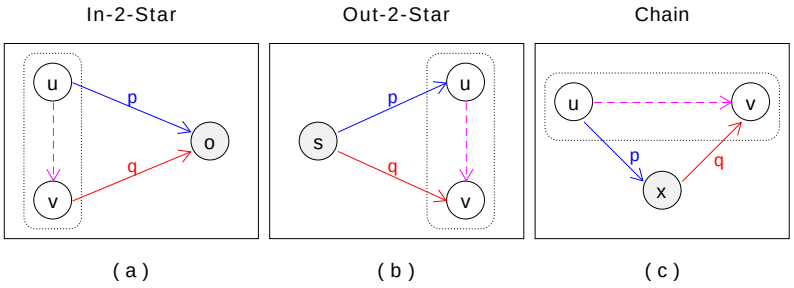

Figure 4. Extraction patterns with two predicates p, q and two triples. (**a**) In-2-Star: two connections are coming-into o from u (using p) and v (using q). (**b**) Out-2-Star: two connections are going-out s to u (using p) and to v (using q). (**c**) Chain: two successive connections between u and v via x, using p and q respectively.

The following SPARQL query can be used to extract all such ties among nodes in the RDF graph using two predicates $p, q \in \mathbf{I}$, and three variables $?u, ?v, ?o \in \mathbf{X}$:

$$Q_8 = Project(\{?u, ?v\}, \{(?u, p, ?o), (?v, q, ?o)\}) \tag{8}$$

It can be translated into SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?u p ?o. ?v q ?o.}
```

In contrast to the in-2-star variant of same-predicate-derivation where the derived relation is bidirectional, here the in-2-star variant of derivation by different predicates is uni-directional, not because of the positions of nodes in the RDF triples (as in direct extraction, or chain variant) but because of the difference of predicates. In fact, one of the predicates can be considered as primary and the other as secondary, for instance, the primary predicate is more important than, or has a priority over, the secondary one. Thus, the derived tie between nodes would be directed *from* the subject of the triple with the primary predicate *to* the subject of the triple with the secondary predicate. Let us clarify this idea with an example.

Example 6. *The YAGO dataset has two predicates* `yago:exports` *and* `yago:imports` *that relate countries to products they export and import, respectively. These predicates can be used to derive a so-called* commercial-dependency *relationship among countries. If we want the relation to be such that a country is linked to another when the former exports a product imported by the later, then we consider* `exports` *predicate as primary, and* `imports` *as secondary, hence, the corresponding query is:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{yago:exports}, ?o), (?v, \texttt{yago:imports}, ?o)\})$$

In this case, the tie from u to v means that u has v as dependent. However, we could instead consider `imports` *as primary and* `exports` *as secondary, here the derived tie from u to v would mean that u depends on v.*

A weighted version of in-2-star variant of derivation using different predicates can be given by the following SPARQL query:

$$Q_9 = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?o\rangle, \{(?u, p, ?o), (?v, q, ?o)\})) \quad (9)$$

which can be syntactically translated into:

```
SELECT ?u ?v (COUNT(?o) AS ?w) WHERE { ?u p ?o. ?v q ?o } GROUP BY ?u ?v
```

6.1.2. Out-2-Star

This pattern is similar to the out-2-star pattern in derivation using same predicate: a tie between two nodes u, v is derived when both nodes have incoming links from a third node s that is, when u and v are in the *object* position of two RDF triples having the same *subject* o. However, the two triples have different predicates p and q, thus the basic graph pattern is: $\{(s, p, u), (s, q, v)\}$.

The following SPARQL query can be used to extract all such ties among nodes in the RDF graph using two predicates $p, q \in \mathbf{I}$, and three variables $?u, ?v, ?s \in \mathbf{X}$:

$$Q_{10} = Project(\{?u, ?v\}, \{(?s, p, ?u), (?s, q, ?v)\}) \quad (10)$$

It can be written in SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?s p ?u. ?s q ?v }
```

Moreover, the BGP can be shortened as: `{?s p ?u; q v?.}` since the triples share the same subject.

In this type of derivation, the extracted network is *directed*, and can be *weighted*. A weighted network can be extracted using this query:

$$Q_{11} = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?s\rangle, \{(?s, p, ?u), (?s, q, ?v)\})) \quad (11)$$

which can be syntactically expressed as:

```
SELECT ?u ?v (COUNT(?s) AS ?w) WHERE { ?s p ?u; q ?v. } GROUP BY ?u ?v
```

Example 7. *The BNB Linked Data Platform (http://bnb.data.bl.uk/) provides access to the British National Bibliography published as linked open data and made available through SPARQL services. Bibliographic resources have creators (authors) and contributors (editors, or publishers, etc.). A resource is related to a creator using the predicate* `dc:creator`*, and to a contributor using* `dc:contributor`*, both predicates come from Dublin Core vocabulary (http://dublincore.org/documents/dcmi-terms/). Thus, a collaboration social network can be derived such that it relates creators of bibliographic resources to other contributors using this query:*

$$Q = Project(\{?u, ?v\}, \{(?s, \texttt{dc:creator}, ?u), (?s, \texttt{dc:contributor}, ?v)\})$$

Example 8. *DBpedia has a* `dbo:director` *predicate that relates a movie to the person who directed it. Thus, along with* `dbo:starring` *predicate, we can derive a network between actors and directors who have worked together on a movie. However, we should specify the direction we desire for the ties; if we want the ties to be* from *directors* to *stars, then the pattern should be:*

$$Project(\{?u, ?v\}, \{(?s, \texttt{dbo:director}, ?u), (?s, \texttt{dbo:starring}, ?v)\}),$$

whereas, if we want it to be from *stars* to *directors, the pattern should be:*

$$Project(\{?u, ?v\}, \{(?s, \texttt{dbo:starring}, ?u), (?s, \texttt{dbo:director}, ?v)\})$$

6.1.3. Chain

This pattern is similar to the *chain* pattern in derivation using the same predicate. A node u is tied to another node v when: (1) u is the *subject* of a triple, (2) v is the *object* of another triple, and (3) x is the *object* of the first triple and the *subject* of the other. However, here the predicates of the triples are different: p and q. Thus, the pattern can be expressed as: $(u, p, x), (x, q, v)$, and the SPARQL query used to extract all such ties is:

$$Q_{12} = Project(\{?u, ?v\}, \{(?u, p, ?x), (?x, q, ?v)\}) \quad (12)$$

which can be expressed in SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?u p/q ?v. } GROUP BY ?u ?v
```

Here, the network is also directed and can be weighted. The query for the weighted version is:

$$Q_{13} = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?x\rangle, \{(?u, p, ?x), (?x, q, ?v)\})) \quad (13)$$

```
SELECT ?u ?v (COUNT(?x) AS ?w) WHERE { ?u p ?x. ?x q ?v. } GROUP BY ?u ?v
```

Example 9. *In DBpedia, we can use the predicates* `dbo:spouse` *and* `dbo:parent`*, to derive a social network where a person is related to his parent-in-law using the following query:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{dbo:spouse}, ?x), (?x, \texttt{dbo:parent}, ?v)\})$$

Example 10. *DBpedia has the predicate* `dbo:employer` *that relates a person (an employee) to an organization where he is employed, and the predicate* `dbo:president` *that relates an organization to its president. Using these two predicates with chain pattern, we can derive a social network that relates employees to their employers:*

$$Q = Project(\{?u, ?v\}, \{(?u, \texttt{dbo:employer}, ?x), (?x, \texttt{dbo:president}, ?v)\})$$

6.2. Patterns with Three Triples

Among different possible patterns of two predicates and three triples, we chose to present two common patterns as shown in Figure 5.

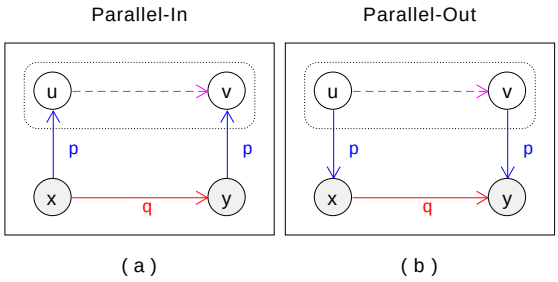

Figure 5. Extraction patterns with two predicates and three triples. (**a**) Parallel-In: u and v have in-coming connections (using p) from x and y respectively. (**b**) Parallel-Out: u and v have out-going connections (using p) to x and y respectively. In both patterns, x is connected to y using q.

6.2.1. Parallel-In

In this pattern, a tie is extracted from a node u to another one v, when both nodes have in-links (they are in object position in two triples with predicate p) from two other intermediary nodes x and y

that are linked to each other using the other predicate q. Thus, the pattern is: $(x, p, u), (y, p, v), (x, q, y)$. Given the two predicates $p, q \in \mathbf{I}$, and four variables $u, v, x, y \in \mathbf{X}$, the corresponding query is:

$$Q_{14} = Project(\{?u, ?v\}, \{(?x, p, ?u), (?y, p, ?v), (?x, q, ?y)\}) \quad (14)$$

which can be syntactically written as:

```
SELECT ?u ?v WHERE { ?x p ?u. ?y p ?v. ?x q ?y. }
```

The extracted relation is directional, such that the direction of the extracted tie (from u to v) is the same direction (parallel) of the relation between the corresponding intermediary nodes (from x to y).

Example 11. *The Bibliographic Ontology (http://bibliographic-ontology.org/) provides main concepts and properties for describing citations and bibliographic references on the Semantic Web. The predicate* `dc:contributor` *relates a document to its author, while* `bibo:cites` *relates a document to another document that cites the first document. Thus, using these predicates, we can extract a citation social network among authors, such that it relates an author to another one when the first writes a document that cites a document written by the second author. This network is extracted using the following query:*

$$Q = Project(\{?u, ?v\}, \{(?x, \texttt{dc:contributor}, ?u), (?y, \texttt{dc:contributor}, ?v), (?x, \texttt{bibo:cites}, ?y)\})$$

A weighted social network can also be extracted using *parallel-in* when we consider the number of co-occurrences of the intermediary relations (?x, q, ?y). The corresponding query is:

$$Q_{15} = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?x, ?y\rangle, \{(?x, p, ?u), (?y, p, ?v), (?x, q, ?y)\})) \quad (15)$$

which can be translated into SPARQL syntax as:

```
SELECT ?u ?v (COUNT(*) AS ?w)
WHERE { ?x p ?u. ?y p ?v. ?x q ?y. }
GROUP BY ?u ?v
```

6.2.2. Parallel-Out

In this pattern, a tie is extracted from a node u to another one v, when both nodes have out-links (they are in subject position in two triples with predicate p) to two other intermediary nodes x and y that are linked to each other using the other predicate q. Thus, the pattern is: $(u, p, x), (v, p, y), (x, q, y)$. The extracted relation is directional. The corresponding query is:

$$Q_{16} = Project(\{?u, ?v\}, \{(?u, p, ?x), (?v, p, ?y), (?x, q, ?y)\}) \quad (16)$$

and the weighted version is:

$$Q_{17} = Project(\{?u, ?v, ?w\}, GroupAgg(\langle ?u, ?v\rangle, ?w, Count, \langle ?x, ?y\rangle, \{((?u, p, ?x), (?v, p, ?y), (?x, q, ?y)\})) \quad (17)$$

Corresponding queries in SPARQL syntax are, respectively:

```
SELECT ?u ?v WHERE { ?u p ?x. ?v p ?y. ?x q ?y. }
```

```
SELECT ?u ?v (COUNT(*) AS ?w)
WHERE { ?u p ?x. ?v p ?y. ?x q ?y. }
GROUP BY ?u ?v
```

6.3. Patterns with Four Triples

Among different possible patterns of two predicates and four triples, we chose to present four common patterns as shown in Figure 6. Networks extracted using these patterns are *undirected* and can be *weighted* when the number of co-occurrences is considered; however, we will not provide weighted versions for the sake of brevity.

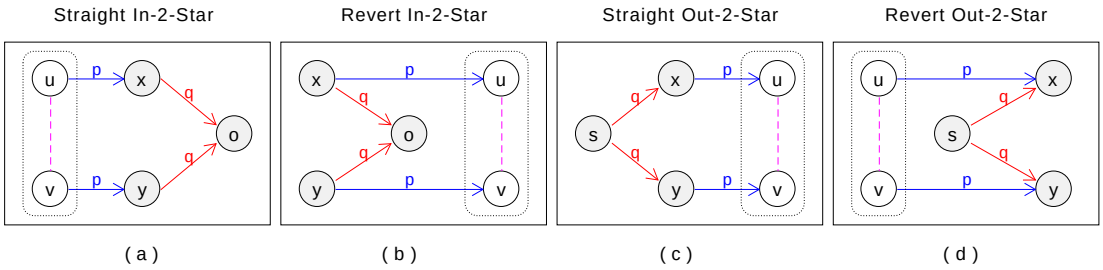

Figure 6. Extraction patterns with two predicates and four triples. (**a**) Straight In-2-Star: u and v have out-going connections (using p) to x and y, respectively, while x and y have out-going connections (using q) to o. (**b**) Revert In-2-Star: u and v have in-coming connections (using p) from x and y, respectively, while x and y have out-going connections (using q) to o. (**c**) Straight Out-2-Star: u and v have in-coming connections (using p) from x and y, respectively, while x and y have in-coming connections (using q) from s. (**d**) Revert Out-2-Star: u and v have out-going connections (using p) to x and y, respectively, while x and y have in-coming connections (using q) from s.

6.3.1. Straight in-2-Star

In this pattern, we extract a bidirectional tie between two nodes u and v if they are linked (using one predicate p) to two other nodes x and y that are in turn linked to a fifth node o using another predicate q (Figure 6a). This pattern $(?u, p, ?x), (?v, p, ?y), (?x, q, ?o), (?y, q, ?o)$ is named so because the intermediary nodes make an in-2-star pattern, and receive straight links from u, v using p (same direction as q). The corresponding query is then:

$$Q_{18} = Project(\{?u, ?v\}, \{(?u, p, ?x), (?v, p, ?y), (?x, q, ?o), (?y, q, ?o)\}) \quad (18)$$

The corresponding syntactic query is written as:

```
SELECT ?u ?v WHERE { ?u p/q ?o. ?v p/q ?o. }
```

Example 12. *In the DailyMed dataset (https://dailymed.nlm.nih.gov/dailymed/), the predicate* `dailymed:producesDrug` *relates a pharmaceutical company to a drug it produces, and the predicate* `dailymed:activeIngredient` *relates a drug to its active ingredient. Using these two predicates, we can extract a social network of competing pharmaceutical companies where competition is defined by selling drugs with the same active ingredient [28]. The required query can be written as:*

$$Q = Project(\{?u, ?v\}, \{ (?u, \texttt{dailymed:producesDrug}, ?x), (?v, \texttt{dailymed:producesDrug}, ?y), (?x, \texttt{dailymed:activeIngredient}, ?o), (?y, \texttt{dailymed:activeIngredient}, ?o) \})$$

where u and v are companies producing x and y drugs, respectively, and the drugs both have the same active ingredient o.

6.3.2. Revert in-2-Star

In this pattern, we extract a bidirectional tie between two nodes u and v if they have links (using one predicate p) from two other nodes x and y that are in turn linked to a fifth node o

using another predicate q (Figure 6b), thus the pattern is: $(?x, p, ?u), (?y, p, ?v), (?x, q, ?o), (?y, q, ?o)$. This pattern is similar to the previous one. However, the links of predicate p are not in the same direction as those of q (hence the name *revert*). The corresponding query is then:

$$Q_{19} = Project(\{?u, ?v\}, \{(?x, p, ?u), (?y, p, ?v), (?x, q, ?o), (?y, q, ?o)\}) \quad (19)$$

The corresponding syntactic query is written as:

```
SELECT ?u ?v
WHERE { ?x p ?u;  q ?o.
?y p ?v;  q ?o. }
```

Example 13. *The Bibliographic Ontology has a predicate* `bibo:presentedAt` *that relates a document to an event—for example, a paper to a conference. Thus, using this predicate and* `dc:contributor` *that relates a document to his author, we can extract a social network among authors whose papers are presented at the same event. The required query can be written as:*

$$Q = Project(\{?u, ?v\}, \{ (?x, \texttt{bibo:presentedAt}, ?o), (?y, \texttt{bibo:presentedAt}, ?o), (?x, \texttt{dc:contributor}, ?u), (?y, \texttt{dc:contributor}, ?v) \})$$

6.3.3. Straight Out-2-Star

In this pattern, we extract a bidirectional tie between two nodes u and v if they have links (using one predicate p) from two other nodes x and y that in turn have links from a fifth node s using another predicate q (Figure 6c). This pattern is denoted $(?x, p, ?u), (?y, p, ?v), (?s, q, ?x), (?s, q, ?y)$, and is named so because the intermediary nodes make an out-2-star pattern, and send straight links out to u, v using p (same direction as q). The corresponding query then is:

$$Q_{20} = Project(\{?u, ?v\}, \{(?x, p, ?u), (?y, p, ?v), (?s, q, ?x), (?s, q, ?y)\}) \quad (20)$$

Syntactically, the query can be written as follows, using a property sequence path q/p:

```
SELECT ?u ?v WHERE { ?s q/p ?u, ?v. }
```

6.3.4. Revert Out-2-Star

In this pattern, we extract a bidirectional tie between two nodes u and v if they are linked (using one predicate p) to two other nodes x and y that in turn have links from a fifth node s using another predicate q (Figure 6d), thus the pattern is: $(?u, p, ?x), (?v, p, ?y), (?s, q, ?x), (?s, q, ?y)$. The corresponding query then is:

$$Q_{21} = Project(\{?u, ?v\}, \{(?u, p, ?x), (?v, p, ?y), (?s, q, ?x), (?s, q, ?y)\}) \quad (21)$$

which can expressed using SPARQL syntax as:

```
SELECT ?u ?v WHERE { ?u p ?x. ?v p ?y. ?s q ?x; q ?y. }
```

To conclude this section, we summarize the extraction patterns for complete networks in Figure 7 where the patterns are classified based on the number of predicates and number of triples.

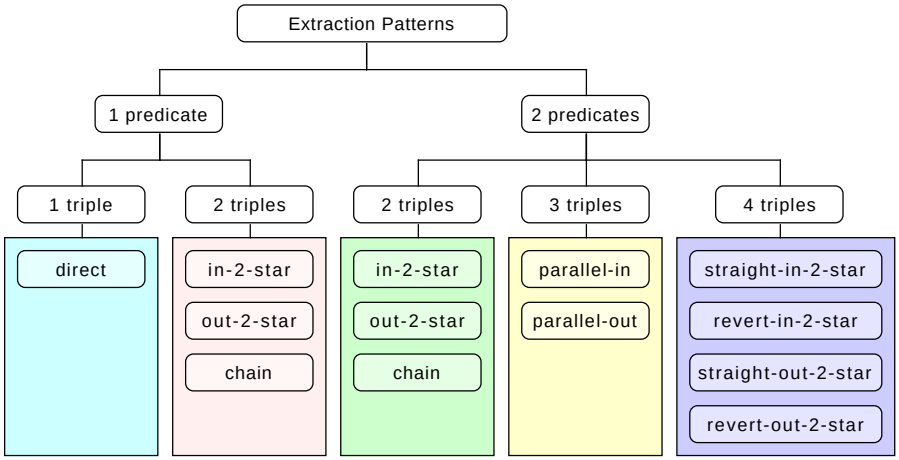

Figure 7. Classification of extraction patterns.

Table 2 also provides an overview of those extraction patterns. Note that, in all these techniques, an extracted tie is always denoted with variables $?u$ and $?v$, and that when the network is directed the direction of the tie is always from $?u$ to $?v$.

The list of extraction techniques/patterns presented so far in this paper is not exhaustive. Many other useful patterns can be added to this list.

Table 2. Summary of extraction techniques for complete networks.

	Variant	Basic Graph Pattern	Extracted Network	
			Directed?	Weighted?
1-triple	direct	$\{(?u, \mathtt{p}, ?v)\}$	yes	no
1-pred., 2-triples	in-2-star	$\{(?u, \mathtt{p}, ?o), (?v, \mathtt{p}, ?o)\}$	no	yes
	out-2-star	$\{(?s, \mathtt{p}, ?u), (?s, \mathtt{p}, ?v)\}$	no	yes
	chain	$\{(?u, \mathtt{p}, ?x), (?x, \mathtt{p}, ?v)\}$	yes	yes
2-pred., 2-triples	in-2-star	$\{(?u, \mathtt{p}, ?o), (?v, \mathtt{q}, ?o)\}$	yes	yes
	out-2-star	$\{(?s, \mathtt{p}, ?u), (?s, \mathtt{q}, ?v)\}$	yes	yes
	chain	$\{(?u, \mathtt{p}, ?x), (?x, \mathtt{q}, ?v)\}$	yes	yes
2-pred., 3-triples	parallel-in	$\{(?x, \mathtt{p}, ?u), (?y, \mathtt{p}, ?v), (?x, \mathtt{q}, ?y)\}$	yes	yes
	parallel-out	$\{(?u, \mathtt{p}, ?x), (?v, \mathtt{p}, ?y), (?x, \mathtt{q}, ?y)\}$	yes	yes
2-pred., 4-triples	straight in-2-star	$\{(?u, \mathtt{p}, ?x), (?v, \mathtt{p}, ?y), (?x, \mathtt{q}, ?o), (?y, \mathtt{q}, ?o)\}$	no	yes
	revert in-2-star	$\{(?x, \mathtt{p}, ?u), (?y, \mathtt{p}, ?v), (?x, \mathtt{q}, ?o), (?y, \mathtt{q}, ?o)\}$	no	yes
	straight out-2-star	$\{(?x, \mathtt{p}, ?u), (?y, \mathtt{p}, ?v), (?s, \mathtt{q}, ?x), (?s, \mathtt{q}, ?y)\}$	no	yes
	revert out-2-star	$\{(?u, \mathtt{p}, ?x), (?v, \mathtt{p}, ?y), (?s, \mathtt{q}, ?x), (?s, \mathtt{q}, ?y)\}$	no	yes

7. Translation into Networks

To this end, we have presented several techniques (as extraction patterns) to extract social networks from Linked Open Data represented using RDF data model. Each of those patterns has the form of a SPARQL query in its algebraic form. As we saw in Section 4.2, queries in SPARQL are expressions of the form $Project(X, P)$ where P is a pattern and X is a set of free variables. The result of a SPARQL query Q over an RDF graph G is a multiset of solution mappings $\mathcal{M} = [\![P]\!]_G$, where $[\![.]\!]_G$

is an evaluation function. In order to obtain the target social network, we need a formal translation step to specify the results of a query (corresponding to an extraction pattern) as a social network. We distinguish between two cases:

7.1. Binary Networks

In this case, the query has two returned variables representing connected vertices. Thus, it has the form: $Q = Project(\{?u, ?v\}, P)$, where P is some pattern (as described in previous section). Let $\mathcal{M} = [\![Q]\!]_D$ be a set of solution mappings $\{\mu\}$ of query Q. To translate $\mathcal{M}$ to a one-mode network: $H(V, E)$, we define a translation function tr: $tr(\mathcal{M}) = H(V, E)$, where

$$V = \{\mu(?u) \mid \mu \in \mathcal{M}\} \cup \{\mu(?v) \mid \mu \in \mathcal{M}\}$$
$$E = \{(\mu(?u), \mu(?v)) \mid \mu \in \mathcal{M}\}$$

7.2. Weighted Networks

In this case, the query has three returned variables: the first two represent connected vertices, and the third is the weight. Thus, the query has the form: $Q = Project(\{?u, ?v, ?w\}, P)$, where P is some pattern. Let $\mathcal{M} = [\![Q]\!]_D$ be a set of solution mappings $\{\mu\}$ of query Q. To translate $\mathcal{M}$ to a one-mode weighted network: $H(V, E, \omega)$, we define a translation function tr: $tr(\mathcal{M}) = H(V, E, \omega)$, where:

$$V = \{\mu(?u) \mid \mu \in \mathcal{M}\} \cup \{\mu(?v) \mid \mu \in \mathcal{M}\}$$
$$E = \{e_\mu = (\mu(?u), \mu(?v)) \mid \mu \in \mathcal{M}\}$$
$$\omega(e_\mu) = \mu(?w)$$

8. Discussion

This paper proposes several techniques to extract social networks from Linked Open Data. The proposed techniques have the form of extraction patterns that can be expressed using SPARQL queries whose results make up the target social network. The importance of the proposed approach comes from (1) the importance of Linked Open Data as a rich source of information, and (2) the role of extraction patterns as guidelines for the process of deriving new latent knowledge (social networks) from existing one (linked open data).

Linked Open Data is structured information in a machine-processable format, openly published on the Web, and linked to other datasets. Those properties of LOD make it an invaluable resource of information, and create new opportunities for many areas of application. Thus, LOD is being increasingly adopted, not only by the scientific community, but also by several groups of stakeholders such as media, industry, and governmental organizations and NGOs. LOD is already widely available in several industries, including libraries, bio-medicine, and government data. "Linking information from different sources is key for further innovation. If data can be placed in a new context, more and more valuable applications—and therefore knowledge—will be generated" [13].

From this point of view comes our proposal of mining new information i.e., social networks, from LOD, and then turning it into knowledge, through social network analysis. Hence, the extraction techniques/patterns proposed in this paper come to facilitate this process.

Extraction patterns can be considered as guidelines to help the user figure out the appropriate formulation of the query to extract a desired network, and to understand the outcomes of different design choices: which predicates are needed, how many triple patterns, which direction of predicate of each triple pattern (subject-object) is the appropriate, etc. Extraction patterns are used as building blocks to establish more complex patterns that can be used to extract complex networks (e.g., as in 3-triple and 4-triple patterns). Moreover, they can also be used as building blocks to design extraction patterns for other types of social networks, such as contextual networks and ego-centered networks.

A contextual social network differs from a complete network in that it covers a subset of the population defined by means of a specific context, e.g., entity type, time, location, or gender (e.g., a co-acting network of Indian actors, or influence network of intellectuals in a specific era). Given the general extraction patterns presented in this paper, specialized extraction patterns for contextual social networks can be constructed by applying additional triple patterns and/or filters that specify the desired context of a target partial network.

On the other hand, an ego-centered network is centered around a specific entity and includes its surrounding environment, e.g., a co-acting network centered around Jodie Foster, or an influence network centered around Isaac Newton. Hence, specialized extraction patterns for ego-centered social networks can be built on top of the general extraction patterns, taking into account whether the network is directed or not, and considering both ego-alter ties and alter-alter ties, as we demonstrated in a previous work [34].

In this paper, the focus has been on the case where a single dataset is being queried at a time, that is, the described patterns have a limited scope to one dataset only (e.g., from a movie subset, or from a bibliographic resource). However, this work can be extended to tackle the case where multiple datasets can be used to extract a target social network. This can be done using the interlinking among LOD datasets, as well as using federated SPARQL queries. One of the main objectives of Linked Open Data is linking and integration among the LOD cloud datasets. "Connectivity among two or more datasets can be achieved through common *Entities*, *Triples*, *Literals*, and *Schema Elements*, while more connections can occur due to equivalence relationships between URIs, such as `owl:sameAs`, `owl:equivalentProperty` and `owl:equivalentClass`, since many publishers use such equivalence relationships, for declaring that their URIs are equivalent with URIs of other datasets" [35]. As most of LOD datasets are interlinked, there are considerable amounts of overlap of RDF resources within datasets in the whole LOD cloud. Thus, such overlap is also reflected onto the social networks extracted from different datasets.

For example, consider the co-acting social network as described in the motivation example (Section 2). This network can be extracted from YAGO dataset using the predicate `yago:actedIn` (which relates an actor to a movie) with the in-2-star extraction pattern (Section 5.3.1) as demonstrated in Example 3. In this case, the size of the network is 225,790 edges, connecting 26,544 nodes (actors).

It is also possible to extract such a network from DBpedia using the predicate `dbo:starring` (which relates a movie to an actor) with the out-2-star extraction pattern (Section 5.3.2) as mentioned in Example 4. The SPARQL query is shown in Figure 8. In this case, the size of the network is 829,887 edges. This network is different from the one extracted from YAGO, not only in terms of the number of entities and edges, but also in terms of the entities themselves (RDF resources), as the entities in YAGO belong to the namespace `http://yago-knowledge.org/resource/`, whereas the entities in DBpedia belong to the namespace `http://dbpedia.org/resource/`.

```
PREFIX dbo: <http://dbpedia.org/property/>
SELECT DISTINCT ?actor1 ?actor2
WHERE {
?movie dbo:starring ?actor1.
?movie dbo:starring ?actor2.
FILTER ( isIRI(?actor1) && isIRI(?actor2)
&& STR(?actor1) > STR(?actor2) )
}
```

Figure 8. SPARQL query to extract co-acting social network from DBpedia.

Despite the differences between the two extracted networks, there are certainly many overlaps between them. For instance, the entity `yago:Brad_Pitt` from yago is the same as the entity `dbr:Brad_Pitt` from DBpedia (Here, the prefix `dbr` refers to DBpedia resources namespace: http://dbpedia.org/resource/). The good news is that, thanks to the interlinking of DBpedia and YAGO,

such equivalences of entities are available via the OWL property `owl:sameAs`. Thus, the overlap between the two co-acting social networks (from YAGO and DBpedia) can be easily detected. Figure 9 shows another version of the previous SPARQL query (to extract the network from DBpedia) where each entity from DBpedia is associated with its equivalent entity from YAGO. The results of this query consist of 94,311 ties/edges that correspond to the intersection of the the two social networks.

```
PREFIX dbo: <http://dbpedia.org/property/>
SELECT DISTINCT ?actor1 ?actor2 ?a1 ?a2
WHERE {
?movie dbo:starring ?actor1.
?movie dbo:starring ?actor2.
?actor1 owl:sameAs ?a1.
?actor2 owl:sameAs ?a2.
FILTER ( isIRI(?actor1) && isIRI(?actor2) && STR(?actor1) > STR(?actor2)
&& strStarts(STR(?a1), ‘‘http://yago-knowledge.org/resource/’’)
&& strStarts(STR(?a2), ‘‘http://yago-knowledge.org/resource/’’)  )
}
```

Figure 9. SPARQL query to extract co-acting network from DBpedia, with YAGO equivalent entities.

It is also possible to perform such an overlap investigation using federated SPARQL queries [16] (through `SERVICE` operator) which allow for combining graph patterns that can be evaluated over several endpoints within a single query [15].

Overall, extracting social networks from linked open data enables us to visualize those networks and study them using prominent tools of social network analysis. Besides usual types of analysis, such as connectivity and centrality, advanced analysis can be applied on extracted social networks, including e.g., community detection, diffusion dynamics, and link prediction, etc. Moreover, being extracted from linked open data, the nodes of an extracted network are LOD entities and thus can be enriched with their attributes that are readily available in the source LOD dataset. This process will turn the extracted network into a content-rich network whose nodes are associated with rich content information. For instance, consider the co-acting network when each actor is associated with extra metadata, such as country, birth date, and gender. As another example, consider the influence network of intellectuals when we associate each node (scholar) with the historical period in which he/she lived; this makes the influence network into a dynamic network and hence enables longitudinal network studies, i.e., to study how a social network develops or changes over time. In all cases, new knowledge is being generated which would be of a great interest.

Author Contributions: Conceptualization, R.G. and J.P.; writing—original draft preparation, R.G.; writing—review and editing, R.G. and J.P.; supervision, J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LOD	Linked Open Data
SNA	Social Network Analysis
RDF	Resource Description Framework
SPARQL	Simple Protocol and RDF Query Language
W3C	World Wide Web Consortium
YAGO	Yet Another Great Ontology
BGP	Basic Graph Pattern
OWL	Web Ontology Language
URI	Uniform Resource Identifier
IRI	Internationalized Resource Identifier

References

1. Ghawi, R.; Pfeffer, J. Mining Social Networks from Linked Open Data. In *Graph-Based Representation and Reasoning*; Endres, D., Alam, M., Şotropa, D., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 221–229.
2. Bizer, C.; Heath, T.; Berners-Lee, T. Linked data—the story so far. *Int. J. Semant. Web Inf. Syst.* **2009**, *5*, 1–22. [CrossRef]
3. Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; Hellmann, S. DBpedia—A Crystallization Point for the Web of Data. *Web Semant.* **2009**, *7*, 154–165. [CrossRef]
4. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; et al. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. *Semant. Web* **2015**, *6*, 167–195. [CrossRef]
5. Mahdisoltani, F.; Biega, J.; Suchanek, F.M. YAGO3: A Knowledge Base from Multilingual Wikipedias. In Proceedings of the CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 4–7 January 2015.
6. Manola, F.; Miller, E. RDF Primer. *W3C Recomm.* **2004**, *10*, 6.
7. Schreiber, G.; Raimond, Y. RDF 1.1 Primer. *W3C Working Group Note* **2014**.
8. Prud'hommeaux, E.; Seaborne, A. SPARQL Query Language for RDF. *W3C Recomm.* **2008**.
9. Pérez, J.; Arenas, M.; Gutierrez, C. Semantics and Complexity of SPARQL. In *The Semantic Web—ISWC 2006*; Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M., Eds.; Springer: Berlin/Heidelberg, Germany 2006; pp. 30–43.
10. Harris, S.; Seaborne, A. SPARQL 1.1 Query Language. *W3C Recomm.* **2013**, *21*, 778.
11. Hennig, M.; Brandes, U.; Pfeffer, J.; Mergel, I. *Studying Social Networks: A Guide to Empirical Research*; Campus Verlag: Frankfurt, France, 2012.
12. Opsahl, T. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Soc. Networks* **2013**, *35*, 159–167. Special Issue on Advances in Two-mode Social Networks. [CrossRef]
13. Bauer, F.; Kaltenböck, M. *Linked Open Data: The Essentials: A Quick Start Guide for Decision Makers*; edition mono/monochrom; Vienna, Austria, 2012. Available online: https://www.reeep.org/LOD-the-Essentials.pdf (accessed on 8 July 2020).
14. Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P.F.; Rudolph, S. OWL 2 Web Ontology Language Primer (Second Edition). *W3C Recomm.* **2012**.
15. Buil-Aranda, C.; Arenas, M.; Corcho, O.; Polleres, A. Federating Queries in SPARQL 1.1: Syntax, Semantics and Evaluation. *J. Web Semant.* **2013**, *18*, 1–17. Special Section on the Semantic and Social Web. [CrossRef]
16. Prud'hommeaux, E.; Buil-Aranda, C. SPARQL 1.1 Federated Query. *W3C Recomm.* **2013**, *21*, 113.
17. Passant, A. dbrec—Music Recommendations Using DBpedia. In *The Semantic Web—ISWC 2010*; Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B., Eds.; Springer: Berlin/Heidelberg, Germeny, 2010; pp. 209–224.
18. Di Noia, T.; Mirizzi, R.; Ostuni, V.C.; Romito, D.; Zanker, M. Linked Open Data to Support Content-Based Recommender Systems. In *I-SEMANTICS '12, Proceedings of the 8th International Conference on Semantic Systems*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 1–8. [CrossRef]
19. De Vocht, L.; Softic, S.; Ebner, M.; Mühlburger, H. Semantically Driven Social Data Aggregation Interfaces for Research 2.0. In Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies; Association for Computing Machinery: New York, NY, USA, 2011. [CrossRef]
20. Razis, G.; Anagnostopoulos, I.; Vafopoulos, M. Semantic Social Analytics and Linked Open Data Cloud. In Proceedings of the 10th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), 2015; pp. 1–6.
21. Anagnostopoulos, I.; Razis, G.; Mylonas, P.; Anagnostopoulos, C.N. Semantic Query Suggestion using Twitter Entities. *Neurocomputing* **2015**, *163*, 137–150. [CrossRef]
22. Razis, G.; Anagnostopoulos, I. Discovering Similar Twitter Accounts Using Semantics. *Eng. Appl. Artif. Intell.* **2016**, *51*, 37–49. [CrossRef]
23. Mika, P. Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks. *Web Semant.* **2005**, *3*, 211–223. [CrossRef]

24. San Martín, M.; Gutierrez, C. Representing, Querying and Transforming Social Networks with RDF/SPARQL. In Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications, Crete, Greece, 31 May–4 June 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 293–307. [CrossRef]
25. Erétéo, G.; Buffa, M.; Gandon, F.; Corby, O. Analysis of a Real Online Social Network Using Semantic Web Frameworks. In *The Semantic Web—ISWC 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 180–195.
26. Erétéo, G.; Gandon, F.; Corby, O.; Buffa, M. Semantic Social Network Analysis. *arXiv* **2009**, arXiv:0904.3701.
27. Ghawi, R.; Schönfeld, M.; Pfeffer, J. Towards Semantic-based Social Network Analysis. In Proceedings of the 14th International IEEE Conference on Signal-Image Technologies and Internet-Based Systems (SITIS 2018), Las Palmas de Gran Canaria, Spain, 26–29 November 2018.
28. Groth, P.T.; Gil, Y. Linked Data for Network Science. *CEUR Workshop Proc.* **2011**, *783*. LISC. CEUR-WS.org.
29. Zehetner, M.A. Social Network Analysis in DBpedia. Master's Thesis, University of Vienna, Wien, Austria, 2010.
30. Wasserman, S.; Faust, K. *Social Network Analysis: Methods and Applications*, 1st ed.; Structural analysis in the social sciences, 8; Cambridge University Press: New York, NY, USA, 1994.
31. Kaminski, M.; Kostylev, E.V.; Cuenca Grau, B. Semantics and Expressive Power of Subqueries and Aggregates in SPARQL 1.1. In Proceedings of the 25th International Conference on World Wide Web, Montreal, QC, Canada, 11–15 April 2016; pp. 227–238. [CrossRef]
32. Kaminski, M.; Kostylev, E.V.; Grau, B.C. Query Nesting, Assignment, and Aggregation in SPARQL 1.1. *ACM Trans. Database Syst.* **2017**, *42*, 1–46. [CrossRef]
33. Ghawi, R.; Petz, C.; Pfeffer, J. 'On the Shoulders of Giants', Analysis of a Social Network of Intellectual Influence. In Proceedings of the Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Granada, Spain, 22–25 October 2019; pp. 248–255. [CrossRef]
34. Ghawi, R.; Schönfeld, M.; Pfeffer, J. Extracting Ego-Centric Social Networks from Linked Open Data. In *WI'19: IEEE/WIC/ACM International Conference on Web Intelligence*; ACM: New York, NY, USA, 2019. [CrossRef]
35. Mountantonakis, M.; Tzitzikas, Y. High Performance Methods for Linked Open Data Connectivity Analytics. *Information* **2018**, *9*, 134. [CrossRef]