An Approach Towards Classifying and Navigating RDF data based on Pattern Structures

Mehwish Alam and Amedeo Napoli

LORIA (CNRS – Université de Lorraine)
BP 239, 54506 Vandoeuvre-les-Nancy, France
{mehwish.alam,amedeo.napoli@loria.fr}

Abstract. With an increased interest in machine processable data, more and more data is now published in RDF (Resource Description Framework) format. This RDF data is present in independent and distributed resources which needs to be centralized, navigated and searched for domain specific applications. This paper proposes a new approach based on Formal Concept Analysis (FCA) to create a navigation space over semantic web data. This approach uses an extension of FCA and takes RDF triples and RDF Schema present on several independent sources and provide centralized access over the data resulting from several resources. Afterwards, SPARQL queries can be posed over this navigation space to access these distributed resources from one platform for information retrieval purposes.

Keywords: Formal Concept Analysis, Navigation Space, Linked Open Data, RDF, RDF Schema, Semantic Pattern Structures.

1 Introduction

With the efforts of Semantic Web community many technologies have been offered for publishing machine-readable data on web. It annotates textual data with meta-data and makes it available in the form of ontologies and RDF graphs. One of the emerging source of such data are published in the form of Linked Open Data (LOD) cloud [2]. As a contrast to textual resources, LOD does not need extensive preprocessing as it is already annotated in the form of entities and relationships.

This structured format leads to other kind of challenges. One of the basic characteristics of Semantic Web is that it follows a *decentralized* publication model, meaning that the RDF graphs are published in several distributed resources, instead of creating one knowledge-base of statements any one can contribute new statements and make it publicly available. These resources have nothing in common except some shared terms. These decentralized graphs should be integrated through machine/software agents to provide domain specific applications. Moreover, external schemas in the form of ontologies or taxonomies

Local Proceedings also appeared in ISBN 978-84-606-7410-8, Universidad de Málaga (Dept. Matemática Aplicada), Spain.

can be linked to these data to make sense based on real world conception. Some of the resources may only consist of ontological information and may not contain the instantial information such as SWRC ontology [11] and some resources may only contain instantial information such as DBLP. The problem of how to provide one platform for accessing several and related semantic web resources to build domain specific applications still persists. This paper focuses on how to link several related heterogeneous RDF resources present on distributed locations containing RDF data and RDF Schema into one space which can further be navigated and searched by the end user. We call it a "navigation space" because it provides centralization over RDF triples and also over RDF Schema belonging to several resources. This space provides navigation and querying over RDF triples as well as RDF Schema.

The current paper introduces a new framework based on Formal Concept Analysis [8] which focuses on how a navigation space is created over RDF data by taking into account an existing RDF Schema to provide search capabilities over distributed and heterogeneous Semantic Web resources. In the current study we only consider a part of RDF Schema which keeps subclass-superclass relation. Other constructs such as sub-property and relation between classes are not considered here. FCA alone may not be able to handle the complex data such as RDF data and RDF Schema as it considers only binary data. To deal with this shortcoming, we employ an extension of FCA, namely Pattern Structures [7]. We propose a new framework called Semantic Pattern Structures, that takes RDF triples and the RDF Schema as an input and produces a "navigation space". This *navigation space* provides centralization over distributed RDF resources and keeps a partially ordered organization of the classes of RDF triples with respect to RDF Schema from data sources which may or may not be part of LOD. The lattice structure allows for querying over RDF graph w.r.t. subject, predicate and object and makes use of the partial order relation between the concepts. This enables simultaneous search over schema and the facts contained in an RDF graph. This navigation space allows querying using well-documented and well-know query language, SPARQL. Several data sources are simultaneously queried through one platform. To date this is the first attempt to deal with RDF graph and RDF Schema using pattern structures.

The paper is structured as follows: section 2 gives the motivating example, section 3 introduces the basic notion of Pattern Structures with an example. Section 4 details the framework of semantic pattern structures for creating schema rich navigation space over RDF data. Section 5 defines the querying mechanism over the obtained navigation space. Section 6 describes the experimental results of the framework. Section 7 discusses the related work while Section 8 concludes the paper.

2 Motivating Example

For instance, if a patient has been prescribed some Cardiovascular Agent (CVA) and after a while he is looking for a replacement for his drug because it causes some allergic condition as a side effect. For finding solution, the doctor should be able to access this information for obtaining the suggestions for replacing patient's drug which is used for the same purpose but does not cause the side effect reported by the patient. All the required information is present in different data sources:

- Drugbank[1] contains information about drugs with their categories and the proteins it targets.
- SIDER[2] contains drug with their specific side effects.
- The category of drugs CVA exists in MeSH [3].
- Allergic conditions which is a class of side effects exists in MedDRA[4].

The information required by the domain expert cannot be obtained by posing a simple query on Google. SPARQL queries over each resource should be written to obtain this answer, which further needs manual work to obtain the required suggestions as the answers are in the form of list. In order to access such kind of information, all the data sets must be accessed simultaneously and there should be a single information space for accessing all the desired information through navigation and simple SPARQL queries.

In the current study, we propose a generic framework that provides one space for accessing several data sources related to some domain. The framework proposed in this study is generic because it can be applied to any domain having related information scattered over several resources. Based on the requirements of the user and the applications, the navigation space can be defined on a limited number of factual RDF triples. For example, in case of a cardiologist, he will only be interested in Cardiovascular Agents or some related categories of drugs. This way the application will include the drugs which are Cardiovascular Agents and its related categories. Only a subset of datasets should be considered according to the requirements of the applications. These specification are described by the domain expert as a starting point for narrowing down the huge cloud of Linked Data present on-line.

3 Preliminaries

3.1 Linked Open Data

Recently, Linked Open Data (LOD) [2] has become a standard for publishing data on-line in the form of Resource Description Framework (RDF)[5] which can further be linked to other data sources published in the same format. The idea underlying RDF is based on storing statements about a particular resource, where each statement is represented as $\langle subject, predicate, object \rangle$. A

[1] http://www.drugbank.ca/

[2] http://sideeffects.embl.de/

[3] http://www.ncbi.nlm.nih.gov/mesh/

[4] http://meddra.org/

[5] http://www.w3.org/RDF/

set of linked statements is referred to as RDF graph which constitutes an RDF triple store. For instance, Table 2 keeps RDF triples for drugs with their side effects and drug categories. The prefixes and full forms of all the abbreviations used in this paper are shown in Table 1. Consider $t1$ i.e., $\langle s_1, p_1, o_1 \rangle$, here s_1 is subject, p_1 is predicate and o_1 is the object. A URI **U** in an RDF graph is a general form of URL. The actual representation of the drug s_1 in the form of URI is ′`http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugs/DB00669`′, which is a URI of the DrugBank provided by University of Mannheim containing all the information about the drug in the form of triples. *DB00669* is the id provided by DrugBank to drug s_1 and the name of the drug is obtained by using the `rdfs:label` predicate defined in RDFS vocabulary[6]. The subject denotes the *resource* and the predicate denotes properties of the resource and defines relationship between the subject and the object. In the rest of the paper we use dereferenced resources i.e., s_1 instead of complete URI.

Practically, we consider RDF data w.r.t. three levels according to the value of predicate.

- An RDF triple is at the *schema level* when the predicate has a reserved value from RDFS vocabulary such as `rdfs:subClassOf, rdfs:Class` (represented as `sc` and `Class` respectively in the rest of the paper). The keyword `rdfs:Class` is used to declare that a set of resources is constituting a class and `rdfs:subClassOf` states that all the instances of one class are the instances of another class. For example, in Table 2, the *schema level* is given by triples $t6, t8$. An example of the tree structure contained in an RDF schema (in the rest of the paper when we use RDF Schema/schema we mean the tree structure created by `rdfs:subClassOf`) related to the drug side effect and their categories is shown in Figures 1 and 2 respectively.
- An RDF triple is at the *type level* if it keeps the mappings from an instance to its class and states that a resource is an instance of a class. The RDF triple is at this level when the predicate has a reserved value from RDF vocabulary such as `rdf:type` (represented as `type` in the rest of the paper). For example, in Table 2, the *type level* is given by triples $t4, t5, t7$.
- An RDF triple is at *factual level* when the predicate is related to the domain knowledge and is not a keyword from RDF vocabulary. For example, in Table 2, the *factual level* is given by $t1, t2, t3$.

Table 2 keeps RDF triples from several resources. The information about the side effect of the drugs shown in triples $t1, t2$ is present on SIDER[7] database which keeps the side effects of the drugs contained on the packaging of the drug. The information about the categories of the drugs shown in triples $t3$ is present on DrugBank[8], which keeps detailed information about each of the drugs and the proteins it targets. The schema level information about the drug side effects shown in $t5, t6$ are present in The Medical Dictionary for Regulatory Activities

[6] http://www.w3.org/TR/rdf-schema/

[7] http://sideeffects.embl.de/

[8] http://www.drugbank.ca/

Abbreviation	Term	Abbreviation	Term
p_1	`http://wifo5-04.informatik.uni-mannheim.de/sider/resource/sider/sideEffect`	p_2	`http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/category`
s_1	Sumatriptan	C_1	Acute Coronary Syndrome
s_2	Tadalafil	C_4	Setevens-Johnsons Disorders
s_3	Theophylline	C_2	Tachycardia
s_4	Ticlopidine	C_3	Urticaria
s_5	Vardenafil	C_5	Erythmia Multiforme
D_1	Fibrinolytic Agents	C_6	Coronary Artery Disorders
D_2	Vasoconstrictor Agents	C_7	Cardiac Arrhythmias
D_3	Vasodilator Agents	C_8	Urticarias
D_4	Fibrin Modulating Agents	C_9	Allergic conditions NEC
D_5	Cardiovascular Agents	C_{10}	Allergic conditions
D_6	Molecular Mechanisms of Pharmacological Action	C_{11}	Cardiac Disorders
D_7	Therapeutic Uses	C_{12}	Immune System Disorders

Table 1: This table keeps the abbreviations of the terms used in the rest of the paper.

tid	Subject	Predicate	Object	Provenance
t1	s_1	p_1	o_1	SIDER
t2	s_1	p_1	C_2	SIDER
t3	s_1	p_2	D_2	DrugBank
t4	s_1	`type`	Drug	DrguBank
t5	o_1	`type`	C_1	MedDRA
t6	C_1	`sc`	C_6	MedDRA
t7	D_2	`type`	D_5	MeSH
t8	D_5	`sc`	D_7	MeSH
⋮	⋮	⋮	⋮	⋮

Table 2: RDF triples for the Drug Domain from several resources.

(MedDRA) Terminology[9] which is the international medical terminology. Finally the schema level information related to the drug category shown in triples $t7, t8$ is present in MeSH (Medical Subject Headings)[10] is a controlled vocabulary thesaurus which along with other tree structures keeps classification of drug categories.

3.2 SPARQL

A standard query language for accessing RDF graphs is SPARQL[11] which mainly focuses on graph matching. A SPARQL query is composed of two parts the head and the body. The body of the query contains the Basic Graph Patterns (it is contained in the WHERE clause of the query). It is composed of complex graph patterns that may include RDF triples with variables, conjunctions, disjunctions and constraints over the values of the variables. These graph patterns are matched against the RDF graph and the matched graph is retrieved and manipulated according to the conditions given in the query. The head of the query is an expression which indicates how the answers of the query should be constructed. For instance, consider a simple SPARQL query `SELECT ?x ?y WHERE {` `?x` p_1 `?y`}, then the basic graph pattern `?x` p_1 `?y` will be matched against the triples containing p_1 as predicate i.e., $t1$ and $t2$ (see Table 2). Thus the answer of the query will be (s_1, o_1), (s_1, C_2).

[9] http://www.meddra.org/

[10] http://www.ncbi.nlm.nih.gov/mesh

[11] http://www.w3.org/TR/rdf-sparql-query/

3.3 Pattern Structures

Formal Concept Analysis [8] can process only binary context, more complex data such as graphs, RDF triples can not be directly processed by FCA. Some of the complex data require scaling for these to be considered as binary data. The concept lattice obtained by a binary context mixes between several types of attributes. Pattern structures [7], an extension of FCA, allows direct processing of such kind of complex data such as RDF triples (as we show in this paper). The pattern structures were introduced in [7].

A pattern structure is a triple $(G, (D, \sqcap), \delta)$, where G is the set of entities[12], $(D, \sqcap)$ is a meet-semilattice of descriptions D and $\delta : G \rightarrow D$ maps an entity to a description. More intuitively, a pattern structure is the set of objects with their descriptions with a similarity operation $\sqcap$ on them which represents the similarity of objects. This similarity measure is idempotent, commutative and associative. If $(G, (D, \sqcap), \delta)$ is the pattern structure then the derivation operators can be defined as:

$$A^{\square} := \bigsqcap_{g \in A} \delta(g) \qquad for\ A \subseteq G$$

$$d^{\square} := \{g \in G | d \sqsubseteq \delta(g)\} \qquad for\ d \in D$$

Now the pattern concept can be defined as follows:

Definition 1 (Pattern Concept). *A pattern concept of a pattern structure "$G, (D, \sqcap), \delta$" is a pair (A, d) where $A \subseteq G$ and $d \in D$ such that $A^{\square} = d$ and $A = d^{\square}$, where A is called the concept extent and d is called the concept intent.*

Finally, the obtained concept lattice is called as *pattern concept lattice*, which keeps partially ordered relations between the pattern concepts.

4 Towards Semantic Pattern Structures

The *decentralization* issue is raised when many entities are contributing statements and making it publicly available. Either they are publishing same data using different vocabularies or related data which is distributed over several resources. To provide centralized access over distributed resources this section discusses a framework called Semantic Pattern Structures. This new framework is called Semantic Pattern Structures (SPS) because it is based on Pattern Structures and it deals with data in the form of triples and it classifies these triples with respect to RDF Schema present in the form of taxonomy. This way the final navigation space also keeps the semantic information i.e., background knowledge. The idea underlying the SPS is to create a concept lattice directly from an RDF graph and use the associated RDF Schema present on Linked Open Data to

[12] We rename an object in Pattern Structures as an *entity* to avoid confusion with the object in an RDF triple.

be accessed within one concept lattice. This concept lattice provides search and navigation capabilities over RDF data as well as over the RDF Schema to answer certain queries of the domain expert. The obtained lattice does not only provides access to several data sources but is also a materialization of the associated RDF Schema.

For creating one navigation space over RDF as well as RDF Schema, the first task is to define RDF triples in terms of patterns structures i.e., specifying the entities, their descriptions and the mapping from entities to description. The second task is to select a suitable RDF Schema associated to these RDF triples from distributed data sources based on domain knowledge. After these two preprocessing steps, we define a similarity measure $\sqcap$ over two descriptions which we generalize to the set of descriptions. After defining the similarity measure, we explain how a semantic pattern concept lattice is built using this similarity measure. Finally, we interpret, provide navigation and querying capabilities over the obtained pattern concept lattice.

4.1 From RDF Triples to Semantic Pattern Structures

Firstly, we represent factual RDF triples about certain domain as entities and their descriptions. According to section 3.3, pattern structures are given as $(G, (D, \sqcap), \delta)$. The descriptions D are termed as semantic descriptions and are denoted as D_s. An RDF triple store containing factual level information consists of statements of the form $\langle s_i, p_j, o_k \rangle$, where $i = \{1, \ldots, n\}, j = \{1, \ldots, q\}, k = \{1, \ldots, m\}$. Then, the set of subjects s_i are mapped to the entities in G. As the set of entities G represent the subjects in triples, we represent it as S.

Regarding the type of each object set which is the range of same predicate a suitable RDF Schema is obtained by locating the terms in the closely related RDF Schema. This selection is done based on domain knowledge. RDF Schema contains many constructs such as property, sub-property etc. along with `sc` and information but in this work we use the RDF Schema predicates such as `type` and `sc` introduced in section 3.1. First, the mapping from object to its type is obtained such as $\langle o_k, \texttt{type}, C_1 \rangle$ i.e., o_k is an instance of class C_1. Each object is then replaced by its corresponding class i.e., o_k is replace by C_1. Now, the considered taxonomy information used for defining the similarity measured is present in the RDF Schema by following the predicates such as $rdfs : subClassOf, skos : broader$ e.g., $\langle C_1, \texttt{sc}, C_6 \rangle$, i.e., C_1 is subclass of C_6. Note that we do not extract the trees, we follows this predicate while defining the similarity in the next section.

For instance, to obtain generic drug categories such as cardiovascular agents or generic side effects such as allergic conditions we further add the class information related to each of the objects in the triples. For example, *(*o_k*,*`type`*,*C_2*)* meaning that o_k is an instance of class C_2. Afterwards, we select all the super classes of the current class in the RDF Schema. For example, for C_2, we have C_7, C_{11} and C_{13}. An RDF Schema for drug side effect and drug category is shown in Figure 1 and Figure 2 respectively.

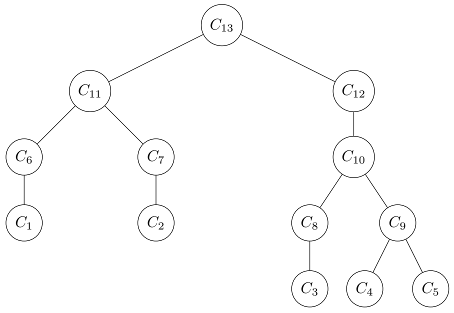

Fig. 1: RDF Schema for Side Effects (MedDRA) ($\mathcal{T}_1$).

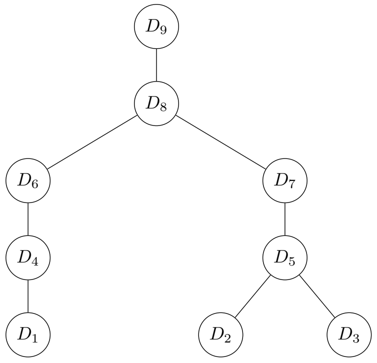

Fig. 2: RDF Schema for Drug Category (MeSH) ($\mathcal{T}_2$).

The pair $(p_j : C_k)$ gives a description $d_{ij} \in D_s$. The mapping of entities to description $\delta : S \rightarrow D_s$ is given as follows: let $s_i \in S$ then $\delta(s_i) = \{d_{i1}, \ldots, d_{iq}\}$ where $i \in \{1, \ldots, n\}$ and $d_{ij} = \{p_j : \{C_1, C_4, \ldots, C_m\}\}$ where $j \in \{1, \ldots, q\}$. Finally, the semantic pattern structures are given as $(S, (D_s, \sqcap), \delta)$. Consider the running scenario described before. As the drugs in the DrugBank and SIDER are same and are the subjects then the triples $t1, t2, t3$ in Table 2 are represented as $(S, (D_s, \sqcap), \delta)$ where the entity S has the description $\delta(s_1) = \{\mathit{(}p_1 : \{C_1, C_2, C_3\}\mathit{)}, \mathit{(}p_2 : \{D_2\}\mathit{)}\ \}$. The descriptions are based on same subjects having different descriptions belonging to different resources i.e., side effect and category. Finally, with respect to the triples in Table 2, the subjects in the RDF triples are mapped to the entities, $S = \{s_1, s_2, \ldots\}$, the descriptions are given as $D_s = \{\mathit{(}p_1 : \{C_1, C_2, C_3, \ldots\}\mathit{)}\ ,\ \mathit{(}p_2 : \{D_1, D_2, \ldots\}\mathit{)}\}$.

Entities S	Descriptions D_s
s_1	$\{(p_1 : \{C_1, C_2, C_3\}), (p_2 : \{D_2\})\}$
s_2	$\{(p_1 : \{C_1, C_4, C_5\}), (p_2 : \{D_3\})\}$
s_3	$\{(p_1 : \{C_2\}), (p_2 : \{D_3\})\}$
s_4	$\{(p_1 : \{C_3, C_4, C_5\}), (p_2 : \{D_8\})\}$
s_5	$\{(p_1 : \{C_1, C_3\}), (p_2 : \{D_2\})\}$

Table 3: RDF Triples as entities S and semantic descriptions D_s.

4.2 Similarity Operation ($\sqcap$) over Descriptions

For defining the similarity operation over the semantic description D_s, we use the notion of least common subsumer (lcs). Here we re-write the definition of a least common subsumer of two classes in a RDF Schema $\mathcal{T}$ given in [9].

Definition 2 (Least Common Subsumer). *Given a partially ordered set $(S, \leqslant)$, a least common subsumer E of two classes C and D (lcs(C,D) for short) in a partially ordered set is a class such that $C \sqsubseteq E$ and $D \sqsubseteq E$ and E is least i.e., if there is a class E' such that $C \sqsubseteq E'$ and $D \sqsubseteq E'$ then $E \subseteq E'$.*

Example 1. Let us consider C_3 and C_5. Then according to the RDF Schema in Figure 1, the common subsumers of C_3 and C_5 are C_{10}, C_{12}, C_{13} and $lcs(C_3, C_5) = C_{10}$.

Similarity Operation over Set of Descriptions: For describing the similarity over set of descriptions, LCS is computed pairwise for classes in two different sets of description and then only the most specific classes are picked.

Let s_1 and s_2 be two entities such that $s_1, s_2 \in S$ and the mapping δ is given as follows $\delta(s_1) = d_1 = \{p_1 : \{C_1, C_2, C_3\}\}$ and $\delta(s_2) = d_2 = \{p_1 : \{C_1, C_4, C_5\}\}$. The similarity is always computed w.r.t the same predicate, here p_1. A dual similarity measure is defined to ensure classification of triples with respect to objects as well as with respect to classes from RDF Schema. The similarity measure is computed based on the taxonomy given in Figure 1. According to the current approach a pairwise similarity measure is computed between the two sets of classes connected through some predicate. For example, $\delta(s_1) \sqcap \delta(s_2) = \langle p_1 : \{C_1, C_2, C_3\}\rangle \sqcap \langle p_1 : \{C_1, C_4, C_5\}\rangle$. Meaning that $\delta(s_1) \sqcap \delta(s_2) = p_1 : \langle lcs(C_1, C_1), lcs(C_1, C_5), \ldots\rangle$. Then, $\delta(s_1) \sqcap \delta(s_2) = p_1 : \{C_1, C_{10}, C_{11}, C_{13}\}$. As we have $C_1 \leqslant C_{11} \leqslant C_{13}$ and $C_{10} \leqslant C_{13}$ then we pick only the specific classes i.e., C_1 and C_{10}. Finally, $\delta(s_1) \sqcap \delta(s_2) = p_1 : \{C_1, C_{10}\}$.

Now let us consider the complete descriptions of the two entities i.e., $\delta(s_1) = \{(p_1 : \{C_1, C_2, C_3\}), (p_2 : \{D_2\})\}$ and $\delta(s_2) = \{(p_1 : \{C_1, C_4, C_5\}), (p_2 : \{D_3\})\}$. Then, $\delta(s_1) \sqcap \delta(s_2) = \{(p_1 : \{C_1, C_2, C_3\}), (p_2 : \{D_2\})\} \sqcap \{(p_1 : \{C_1, C_4, C_5\}), (p_2 : \{D_3\})\}$. The set of least common subsumers for p_1 will be C_1, C_{10}. The least common subumer between $p_2 : \{D_3\}$ and $p_2 : \{D_2\}$ is $p_2 : \{D_5\}$. Finally, the similarity between these two drugs will be $\{(p_1 : \{C_1, C_{10}\}), (p_2 : \{D_5\})\}$.

The proposed similarity measure fulfills the property of a similarity operator i.e., commutative, associative and idempotent [7]. The number of RDF Schema considered are equal to number of predicates in the entity-description representation of an RDF graph. In the running example, we use two schemas to show that our method can also be easily applied to multiple RDF Schemas which are either independent or parts of the same schema distributed over several resources.

4.3 Building the Semantic Pattern Concept Lattice

For building the semantic pattern concept lattice using the above similarity measure over several RDF Schemas, according to the running example, $\delta(s_1) \sqcap \delta(s_2) = \{(p_1 : \{C_1, C_{10}\}), (p_2 : \{D_5\})\}$. The semantic pattern concept lattice is built according to definition 1. For instance, $\{s_1, s_2\}^{\square} = \{(p_1 : \{C_1, C_{10}\}), (p_2 : \{D_5\})\}$ but $\{(p_1 : \{C_1, C_{10}\}), (p_2 : \{D_5\})\}^{\square} = \{s_1, s_2, s_5\}$. This way we obtain $K\#2$ (see Figure 3) with 3 drugs giving the concept a support of 3. Finally, the pattern concept lattice is built for Table 3 with respect to $\mathcal{T}_1$(Figure 1) and $\mathcal{T}_2$(Figure 2). This pattern concept lattice is shown in Figure 3 and the details of each of the pattern concept is shown in Table 4.

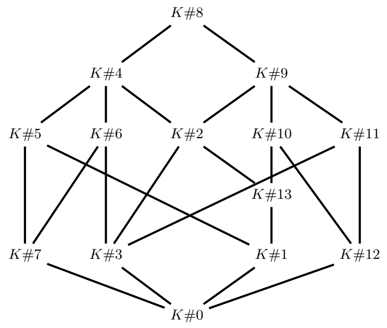

Fig. 3: Pattern Concept lattice for Drugbank

$K\#ID$	**Extent**	**Intent**
$K\#1$	$\{s_1\}$	$(p_1 : \{C_1, C_2, C_3\})$, $(p_2 : \{D_2\})$
$K\#2$	$\{s_1, s_2, s_5\}$	$(p_1 : \{C_1, C_{10}\})$, $(p_2 : \{D_5\})$
$K\#3$	$\{s_2\}$	$(p_1 : \{C_1, C_4, C_5\})$, $(p_2 : \{D_3\})$
$K\#4$	$\{s_1, s_2, s_3, s_5\}$	$(p_1 : \{C_{11}\})$, $(p_2 : \{D_5\})$
$K\#5$	$\{s_1, s_3\}$	$(p_1 : \{C_2\})$, $(p_2 : \{D_5\})$
$K\#6$	$\{s_2, s_3\}$	$(p_1 : \{C_{11}\})$, $(p_2 : \{D_3\})$
$K\#7$	$\{s_3\}$	$(p_1 : \{C_2\})$, $(p_2 : \{D_3\})$
$K\#8$	$\{s_1, s_2, s_3, s_4, s_5\}$	$(p_1 : \{C_{13}\})$, $(p_2 : \{D_8\})$
$K\#9$	$\{s_1, s_2, s_4, s_5\}$	$(p_1 : \{C_{10}\})$, $(p_2 : \{D_8\})$
$K\#10$	$\{s_1, s_4, s_5\}$	$(p_1 : \{C_3\})$, $(p_2 : \{D_8\})$
$K\#11$	$\{s_2, s_4\}$	$(p_1 : \{C_4, C_5\})$, $(p_2 : \{D_8\})$
$K\#12$	$\{s_4\}$	$(p_1 : \{C_3, C_4, C_5\})$, $(p_2 : \{D_9\})$
$K\#13$	$\{s_1, s_5\}$	$(p_1 : \{C_1, C_3\})$, $(p_2 : \{D_2\})$

Table 4: Details of *Pattern Concept lattice* in Figure 3

4.4 Interpreting and Navigating the Pattern Concept Lattice

This pattern concept lattice not only serves as a navigation space but also provides clustering of RDF triples w.r.t. some Schema. Each of the pattern concepts is a cluster of triples. For instance, consider the simplest example in Table 4 i.e., $K\#1$, where there is only one entity in the extent, then this cluster contains the triples $\{(s_1, p_1, C_1), (s_1, p_2, D_2), \dots\}$. Here it can be seen that all the classes are connected to single subject S through two predicates. This space can further be navigated to provide the required information to the user without the technical background of Semantic Web. Let us consider, if user is looking for drugs causing side effect C_{10}, the located pattern concept will be $K\#9$ which contains all the drugs causing some allergic conditions. Afterwards, if the user is looking for some very specific allergic condition such as C_3 then he can simply navigate downwards to obtain more specific concept i.e., $K\#10$ which contains all the drugs which cause the side effect C_3. Let us consider the scenario described in section 2, here the user will be looking for cardiovascular agents (D_5) which do not cause any allergic conditions. In this case first the concept containing all the cardiovascular agents are located i.e., $K\#4$ afterwards the lattice is further navigated downwards to obtain the drugs causing C_{10} (allergic conditions) i.e., $K\#2$. Now the drugs which are in $K\#4$ and not in $K\#2$ are the CVAs which do not cause allergic conditions.

5 SPARQL Queries over Pattern Concept Lattice

For the user with the basic knowledge of SPARQL queries, this *navigation space* can be easily accessed by posing simpler queries. This sub-lattice gives additional information related to the query (detailed below). The obtained pattern concept lattice keeps the materialized RDF Schema meaning that it does not require the query to perform reasoning. It also allows for accessing several resources from one platform. This *navigation space* allows queries which are subject, predicate or object bound. In this section we describe simple as well as complex SPARQL queries that can be posed over the this navigation space. Simple queries refer to the queries with simple Basic Graph Patterns such a single triple pattern in the `WHERE` clause of the SPARQL query, on the other hand, complex queries allow joins and negations.

5.1 Simple Queries

Simple queries include subject, predicate or object bound queries. *Subject-bound queries* refer to the queries where the value of the subject in an RDF triple $\langle s, p, o\rangle$ is known while the values of predicate, object and classes of objects are to be retrieved. Let a SPARQL query be `SELECT ?p ?o WHERE` $\{s_i$ `?p ?o`$\}$, where s_i is a known subject and a pattern concept be (A, d) such that $s_i \in A$. The BGP in this query is (s_i `?p ?o`), this BGP is matched against the RDF graph present in the pattern concept lattice. The answer of the query will be the object concept $\{s_i\}^{\square} = \{d_i\}$ where $d_i \in d$ where d is the intent of the concept (A, d). Note that this object concept is already computed while building a pattern concept lattice. In other words, all the predicate-object pairs $\{d_i\}$ connected to the subject s_i in the RDF graph $\mathcal{G}$ are returned as a result. For example, if user wants to know complete information about the drug s_1 i.e., its side effect and the category, then `SELECT ?p ?o WHERE` $\{s_1$ `?p ?o`$\}$. It will search for object concept in the pattern concept lattice given in Figure 3. For instance, $s_1^{\square} = (p_1 : \{C_1, C_2, C_3\}, p_2 : \{D_2\})$ $(K\#1)$ and the answer would be $(p_1 : \{C_1, C_2, C_3\}, p_2 : \{D_2\})$. The first part of the answer $(p_1 : \{C_1, C_2, C_3\}$ belongs to SIDER while $p_2 : \{D_2\}$ belongs to DrugBank. However, because of the strong lattice-structure a sub-lattice is retrieved by navigating to more general concepts (upward navigation) present in the pattern concept lattice i.e., all the concepts connected to $K\#1$ i.e., $K\#2, K\#4, K\#5, K\#9, K\#10, K\#13, K\#8$. Here, $K\#1$ is the *focus concept* because it keeps the exact answer to the posed

SPARQL query and the rest of the concepts contain additional information. This way the user not only gets the desired answer but also some additional information about the drug i.e., the classification of side effects from MedDRA and the features it shares with other drugs and with which drugs it shares certain features. Object-bound queries are answered in the same way as described above by retrieving the attribute concept and the linked concepts. However, in this case the additional information will be retrieved by navigating downwards from the focus concept because it is the object-bound query and objects are contained in the intent of the pattern concepts.

5.2 Queries with Joins

Now let us consider a slightly more complex query which includes a join and a class resources to be retrieved. If it is specified that the drugs to be retrieved should be Cardiovascular Agent causing some Allergic Condition. Then the SPARQL query with subject-subject join will be as follows:

SELECT ?drug WHERE $\{$
?drug p_2 D_5.
?drug p_1 C_{10} $\}$

For obtaining the answer, the BGP in the above query is matched against the concept which contains both the predicates p_1 and p_2, along with this predicate it keeps the classes of objects `CardiovascularAgents` (D_5) connected to predicate p_2 as well as `AllergicConditions` (C_{10}) connected to the predicate p_1. The answer will be $\{s_1, s_2, s_5\}$ i.e., $K\#2$, which will be the focus concept. However, to obtain more specific information regarding these classes of drug categories and the side effects will be returned by navigating downwards in the concept lattice. Finally, a sub-lattice containing $K\#2, K\#3, K\#13, K\#1$ is returned as an answer to the above query. In this case, the user will have an additional information regarding more specific categories related to Cardiovascular Agents causing Allergic Conditions such as according to K#13 drugs $\{s_1, s_5\}$ are D_2 and K#3 $\{s_2\}$ is D_3. Similarly, specific allergic conditions are also retrieved.

5.3 Queries with Filtering Criteria

Now let us move towards the scenario where the doctor is looking for drug replacement for the patient having heart conditions based on its side effects. For replacing the drugs, the system can be queried by locating all the CVAs (this information comes from DrugBank, MeSH in the current navigation space) and then excluding all the CVAs causing Allergic Conditions. The related SPARQL query can be given as follows:

SELECT ?drug WHERE $\{$
?drug p_2 D_5.

```
?drug $p_1$ ?se
FILTER (?se != $C_{10}$) }
```

The condition is given in the filter clause because SPARQL does not support negation operation directly. In order to do so, all the drugs which are CVAs are retrieved by SPARQL and then it filters the drugs which do not cause allergic conditions. Same is the case with the lattice operation where concept K#4 is selected because it contains all the cardiovascular agents and then a more specific concept is obtained containing the CVAs causing allergic conditions i.e., $K\#2$ and minus operator is applied to perform the desired filtering. The answer obtained is s_3.

In a sense, the navigation space follows the query mechanism provided by Google where a query is posed by the user and this query is mapped to a preexisting cluster of documents. In the same way, when a query is posed by the user, our query mechanism maps the query to a cluster of RDF triples, where each element in an RDF triple keeps the URI of a web-page.

6 Experimentation

The current approach was applied to biomedical data. This section details the experimental evaluation for the creation of the *navigation space* The proposed algorithm was coded in C++ and the experiments was performed using 3GB RAM on Ubuntu version 12.04.

6.1 Drug Search

The datasets containing information about drugs are DrugBank, SIDER, MedDRA and MeSH as described in section 3.1. *DrugBank* keeps detailed information about each of the drugs, their categories and the proteins it targets. The other database is SIDER which keeps the side effects of the drugs contained on the packaging of the drug. The access to these two data sets is provided by University of Mannheim through two different endpoints[13][14].

The schema associated to the side effects of the drug is available on *BioPortal* [12], which is a web portal that provides access to a repository of biomedical ontologies. *BioPortal* is developed by National Center for Biomedical Ontology (NCBO) [10]. The Medical Dictionary for Regulatory Activities (MedDRA) Terminology is the international medical terminology. During this experiment we will enrich side effects with schema level information using MedDRA terminology. In case of the drug categories MeSH vocabulary thesaurus was taken into account. MeSH (Medical Subject Headings) is a controlled vocabulary thesaurus. The drug categories from DrugBank will be enriched with the tree numbers from MeSH vocabulary. The tree numbers arrange the terms from MeSH in a hierarchical manner known as *MeSH Tree Structures.* In the current experiment we

[13] http://wifo5-04.informatik.uni-mannheim.de/drugbank/snorql/

[14] http://wifo5-04.informatik.uni-mannheim.de/sider/snorql/

used the MeSH vocabulary already present in the form of RDF in Bio2RDF [1, 5], which makes the public databases related to Bioinformatics such Kegg, MeSH, HGNC etc. available in the RDF format.

During this experiment, two subsets of the dataset were considered. Both belonging to two major classes of drugs i.e., Cardiovascular Agents (CVA) and Central Nervous System (CNS).

6.2 Evaluation

In the following, we study the scalability of *Semantic Pattern Structures* over large dataset. Table 5 precises the statistics of the data. Pattern concept lattices over both the chosen data sources was built in 0-25 seconds for the maximum of 31098 triples. Figure 4b shows the variation in the size of the navigation space for both data sets. The navigation space contains a maximum of around 500000 clusters of triples which were generated in 25 seconds. However, there are several ways to reduce these number of concepts. The filtering based on the depth of classes considered in the taxonomy which allows the reduction in the number of clusters while generating the concept lattice and hence causes decrease in the runtime of creating the navigation space. Most of these very general classes are not very interesting for the domain expert. Moreover, there are several measures such as support, stability and lift which allow post-filtering of the navigation space.

Datasets	**No. of Triples**	**No. of Subjects**	**No. of Objects**	**Runtime**
Cardiovascular Agents	31098	145	927	0-22 sec
Central Nervous System	22680	105	1050	0-25 sec

Table 5: Statistics of two datasets and navigation space.

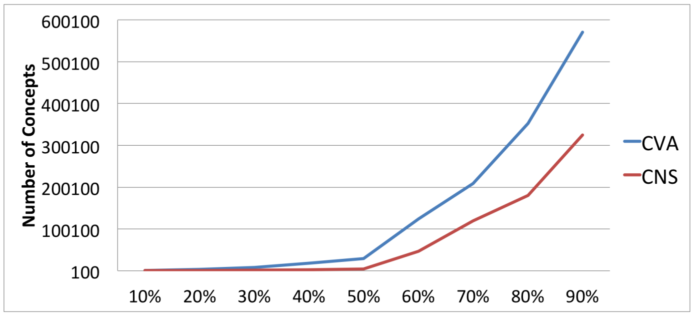

Fig. 4: Size of the Navigation Space for each dataset.

7 Related work

In [6], the author focuses on allowing conceptual navigation to RDF graphs, where each concept is accessed through SPARQL-like queries. However, in our case several RDF graphs are considered and we use the already existing, well-established and well-documented query language, SPARQL. Moreover, [3] introduces ontological pattern structures for enriching raw data with $\mathcal{EL}$ ontologies. But both the approaches consider only one resource at a time, hence not targeting the problem of decentralization. As a contrast, our approach provide navigation space over RDF graphs as well as schema level information from several resources allowing user to access information from one platform. In [4], the authors introduce a new clause `Categorize By` which clusters SPARQL query answers w.r.t to background knowledge present as ontologies. Like our approach, only taxonomy is used for enriching, however, unlike our approach if clusters the answers after obtaining the query answers. As a contrast, our approach provides classification of the RDF triples as well as RDF Schema before hand. Afterwards, SPARQL queries are mapped to the existing clusters and the answer is shown to the user. In such a case, the query response time is faster than the `Categorize By` clause. Moreover, as it provides clustering over answers only, it lacks the capability to provide user with additional information.

8 Conclusion and Future Work

This paper proposes a new approach for navigating semantic web data and targets the capabilities of FCA to deal with RDF data. It provides navigational and search capabilities over RDF triples as well as RDF Schema distributed over several resources. This paper proposes a new similarity measure for pattern structures to deal with RDF data as well as RDF Schema simultaneously, termed as Semantic Pattern Structures. The pattern concepts in the concept lattice are considered as clusters of of RDF triples which can then be navigated and queried by the user.

References

1. François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706–716, 2008.
2. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
3. Adrien Coulet, Florent Domenach, Mehdi Kaytoue, and Amedeo Napoli. Using pattern structures for analyzing ontology-based annotations of biomedical data. In *11th International Conference on Formal Concept Analysis,*, 2013.
4. Claudia d'Amato, Nicola Fanizzi, and Agnieszka Lawrynowicz. Categorize by: Deductive aggregation of semantic web query results. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *ESWC (1)*, volume 6088 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 2010.

5. Michel Dumontier, Alison Callahan, Jose Cruz-Toledo, Peter Ansell, Vincent Emonet, François Belleau, and Arnaud Droit. Bio2rdf release 3: A larger, more connected network of linked data for the life sciences. In *Posters & Demonstrations Track ISWC.*, 2014.
6. Sébastien Ferré. Conceptual navigation in RDF graphs with sparql-like queries. In *Formal Concept Analysis, 8th International Conference, ICFCA 2010, Agadir, Morocco, March 15-18, 2010. Proceedings*, pages 193–208, 2010.
7. Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In Harry S. Delugach and Gerd Stumme, editors, *ICCS*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2001.
8. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.
9. Ralf Küsters and Ralf Molitor. Computing least common subsumers in ALEN. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI*, pages 219–224, 2001.
10. Mark A. Musen, Natalya Fridman Noy, Nigam H. Shah, Patricia L. Whetzel, Christopher G. Chute, Margaret-Anne D. Storey, and Barry Smith. The national center for biomedical ontology. *JAMIA*, 19(2):190–195, 2012.
11. York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The swrc ontology - semantic web for research communities. In *EPIA*, volume 3808 of *Lecture Notes in Computer Science*, pages 218–231. Springer, 2005.
12. Patricia L. Whetzel, Natalya Fridman Noy, Nigam H. Shah, Paul R. Alexander, Csongor Nyulas, Tania Tudorache, and Mark A. Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39(Web-Server-Issue):541–545, 2011.